Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

TRABAJO FINAL DE GRADO

TÍTULO DEL TFG: Seguimiento y control de corredores en carreras populares usando dispositivos Bluetooth Low Energy

TITULACIÓN: Grado en Ingeniería de Sistemas de Telecomunicación

AUTOR: Víctor Dorado Fernández

DIRECTOR: David Pérez Díaz de Cerio

FECHA: 10 de Julio del 2019

Título: Seguimiento y control de corredores en carreras populares usando dispositivos Bluetooth Low Energy

Autor: Víctor Dorado Fernández

Director: David Pérez Díaz de Cerio

Data: 10 de Julio del 2019

Resumen

Este proyecto consiste en implementar una plataforma de seguimiento de corredores en carreras populares complementaria a la ya habitual basada en RFID. La solución consiste en utilizar un dispositivo BLE situado en el dorsal de cada corredor, el cual transmite los datos necesarios a dispositivos móviles que están en posesión de organizadores de la carrera. El dispositivo móvil es el responsable de cargar estos datos en un servidor para que estos puedan ser mostrados a través de una interfaz web.
Por lo tanto, el proyecto consta de tres partes: programación y control de dispositivos BLE, diseño de la aplicación móvil para la recepción de datos y su posterior envío al servidor y por último, gestión del servidor web para la visualización de los datos recogidos.

Title: Tracking and control of runners in popular races using Bluetooth Low Energy devices

Author: Víctor Dorado Fernández

Director: David Pérez Díaz de Cerio

Date: July 10, 2019

Overview

This project consists of implementing a platform for tracking runners in popular races complementary to the usual one based on RFID. The solution is to use a BLE device located on the race number of each runner, which transmits the necessary data to mobile devices that are in possession of race organizers. The mobile device is responsible for uploading this data to a server so that they can be displayed through a web interface.
Therefore, the project consists of three parts: programming and control of BLE devices, design of the mobile application for the reception of data and its subsequent sending to the server and finally, management of the web server for the visualization of the collected data.

Quiero dar las gracias a mi tutor David y a José Luis por toda la ayuda,
al departamento de TSC,
y a todas las personas que me han apoyado.

ÍNDICE

INTRODUCCIÓN

En los tiempos actuales la tecnología inalámbrica es una de las grandes demandas de los usuarios y es por eso que existe una asociación directa entre esta necesidad y la tecnología Bluetooth debido a que es la más utilizada para este tipo de funciones de corto alcance y baja densidad de dispositivos activos.

Sin embargo, Bluetooth es mucho más que eso. Y es gracias al auge del Internet of Things (IoT) que la tendencia de encasillamiento de la tecnología Bluetooth está cambiando y cada vez cobra más importancia.

La revolución del IoT consigue la conexión de millones de dispositivos y servicios diferentes, creando una innumerable cantidad de nuevas aplicaciones en diferentes campos tales como: automoción, salud, casas inteligentes, etc.

Aunque existe una gran parte de dispositivos dentro de IoT que utilizarán sistemas inalámbricos celulares (LTE-M, NB-IoT, 5G), la mayoría de los dispositivos IoT no los utilizarán; en su lugar, emplearán tecnologías que operan en bandas sin licencia de Industrial Scientific Medical (ISM), como Wi-Fi, ZigBee, RFID, Bluetooth y otras, que formarán lo que se conoce como redes capilares.

Hoy en día la tecnología más habitual para realizar el seguimiento de corredores en carreras es la basada en RFID. Los tiempos son calculados a partir de etiquetas RFID pasivas colocadas en los dorsales las cuales son leídas por lectores RFID instalados a lo largo del recorrido de la carrera. El hecho de utilizar esta tecnología implica un aumento del coste en la infraestructura de la carrera ya que la colocación de los lectores es compleja y se han de posicionar debajo de badenes o cualquier otro tipo de estructura.

La idea de este proyecto es crear una alternativa a este sistema utilizando la tecnología BLE, la cual es mucho más accesible y barata debido a su compatibilidad con la mayoría de Smartphones.

El objetivo principal del proyecto consiste en aprovechar los voluntarios de la carrera de manera que se puedan recoger datos utilizando sus propios móviles y de esta manera poder ofrecer una mayor cantidad de datos de tiempos además de poder utilizar la señal GPS de los móviles para obtener una localización más exacta del punto de control y de esta manera poder escenificar el recorrido en un mapa.

CAPÍTULO 1. TECNOLOGÍAS RFID Y BLE

1.1. Tecnología RFID

La tecnología RFID es una de las tecnologías más importantes en el ámbito del internet de las cosas (IoT), como veremos a continuación y por eso vamos a profundizar en ella. Una de sus posibles utilidades es el cronometraje de todo tipo de carreras.

1.1.1. Introducción al RFID

RFID o identificación por radiofrecuencia (Radio Frequency Identification) es un sistema de almacenamiento y recuperación de datos remotos. Su objetivo principal es transmitir la identidad de un objeto mediante radiofrecuencia.

Su coste y su radio de trabajo dependen de la frecuencia en la que se trabaje. Si se trabaja en alta frecuencia (HF), el radio de trabajo es de unos 90 cm. Por otro lado, si se trabaja a ultra alta frecuencia (UHF), permite tener un radio de trabajo de unos 10m, utilizando etiquetas RFID pasivas.

El sistema se basa en tres componentes principales: etiquetas RFID, lectores y subsistema de procesamiento de datos.

La etiqueta RFID (tag RFID) es el elemento que contiene la información y hay tres tipos muy diferenciados:

- Etiquetas RFID activas: Poseen una batería que suministra energía a la antena y el circuito integrado. Permiten transmitir la información que contienen, además del nivel de la batería.
- Etiquetas semi-pasivas: Poseen una batería que suministra energía sólo al circuito integrado.
- Etiquetas pasivas: Poseen una antena que genera la suficiente energía como para activar el circuito integrado y enviar la información. Estas son las más interesantes ya que al no precisar de alimentación por batería u otro tipo de fuente, su tamaño puede ser reducido hasta niveles de poder incluirla en pegatinas, tarjetas, pulseras, etc.

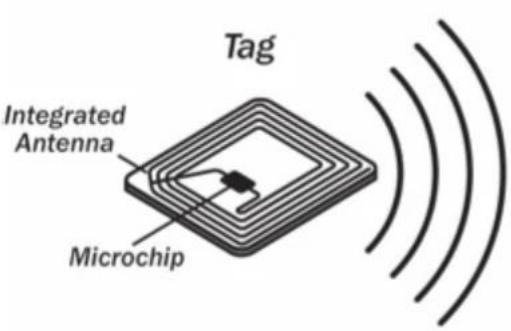

Figura 1.1 Etiqueta RFID

El lector RFID es el encargado de enviar señales para poder detectar las etiquetas en sus cercanías. Cuando detecta la señal de una etiqueta, extrae su información. Otra de sus funciones es transmitir energía a la etiqueta RFID en el caso de que esta sea pasiva además de que también puede escribir datos en la etiqueta.

El subsistema de procesamiento de datos, es el que lleva a cabo el almacenamiento y tratado de la información obtenida por los lectores.

1.1.2. RFID en carreras

La tecnología RFID es hoy en día ampliamente utilizada en el cronometraje de todo tipo de carreras, maratones, carreras ciclistas, etc. En este tipo de eventos se utilizan tags RFID especiales, bien sea para colocar en dorsales o zapatillas, así como lectores RFID colocados en badenes en el suelo o arcos en los puntos de salida o llegada.

En la Figura 1.2 podemos ver un ejemplo de un lector RFID en una carrera, en este caso los lectores estarían situados dentro del badén y captarían las señales de los corredores que pasen por encima.

Gracias a la alta frecuencia de lectura que tiene la tecnología, no es ningún problema para ella el hecho de que pasen varios corredores a la vez ya que deberían ir a una velocidad muy elevada para que no sea detectado por el lector.

Además, el sistema puede ser sincronizado con el disparo de grabación de vídeo con cámaras de alta velocidad, con el fin de obtener elementos de discernimiento en llegadas en masa.

Fig 1.2 Lector RFID

1.2. Tecnología Bluetooth Low Energy

Bluetooth Low Energy (BLE) es una tecnología wireless disponible a partir del estándar BT 4.0. Uno de los grandes puntos fuertes de esta tecnología es el hecho de la aceptación y compatibilidad obtenida por parte de las grandes plataformas como Android, iOS, Microsoft o Linux. Otros puntos fuertes son: tamaño reducido de los chipsets, requerimientos de potencia muy bajos y un aceptable alcance en las comunicaciones.

1.2.1. Introducción al BLE

La tecnología BLE fue introducida por primera vez en la versión 4.0 de Bluetooth. BLE es capaz de utilizar hasta 40 canales de 2MHz en la banda ISM de 2,4 GHz. El estándar emplea la técnica “frequency hopping” o “saltos en frecuencia”, siguiendo una secuencia de saltos pseudo-aleatorios entre los canales frecuenciales mencionados que ofrece un alto grado de robustez frente a interferencias. BLE utiliza Gaussian Frecuency Shift Keying (GFSK) como modulación con el objetivo de simplificar la complejidad del transceptor [1].

Aunque el Bluetooth estándar permite velocidades de transmisión altas, los dispositivos compatibles con la tecnología BLE, permiten una velocidad de hasta 1 Mbps de transmisión.

Los dispositivos que transmiten paquetes de advertising en los canales físicos de advertisement son llamados advertiser, mientras que los dispositivos que están en el modo de escaneo, alerta a la recepción de cualquier advertisement, son llamados Scanner. Como se puede observar en la Figura 1.3, el advertiser genera periódicamente advertisements no conectables y unidireccionales, los cuales consisten en una secuencia de paquetes en los tres canales de advertisement 37, 38, 39.

Los eventos de advertisement sucesivos están separados por un advertisement interval predefinido. Para evitar colisiones repetidas entre dispositivos de advertisement, se agrega un retraso aleatorio al advertisement interval. El Scanner ingresa en el modo de escaneo en intervalos de tiempo periódicos llamados scan interval y escucha los mensajes de advertisement durante un tiempo fijo llamado scan window. El escáner cambia su canal de escaneo después de cada scan interval.

Según el tipo de paquete de advertisement, el Scanner puede realizar una solicitud al anunciante en el mismo canal PHY de advertisement, que puede ir seguido de una respuesta del advertiser.

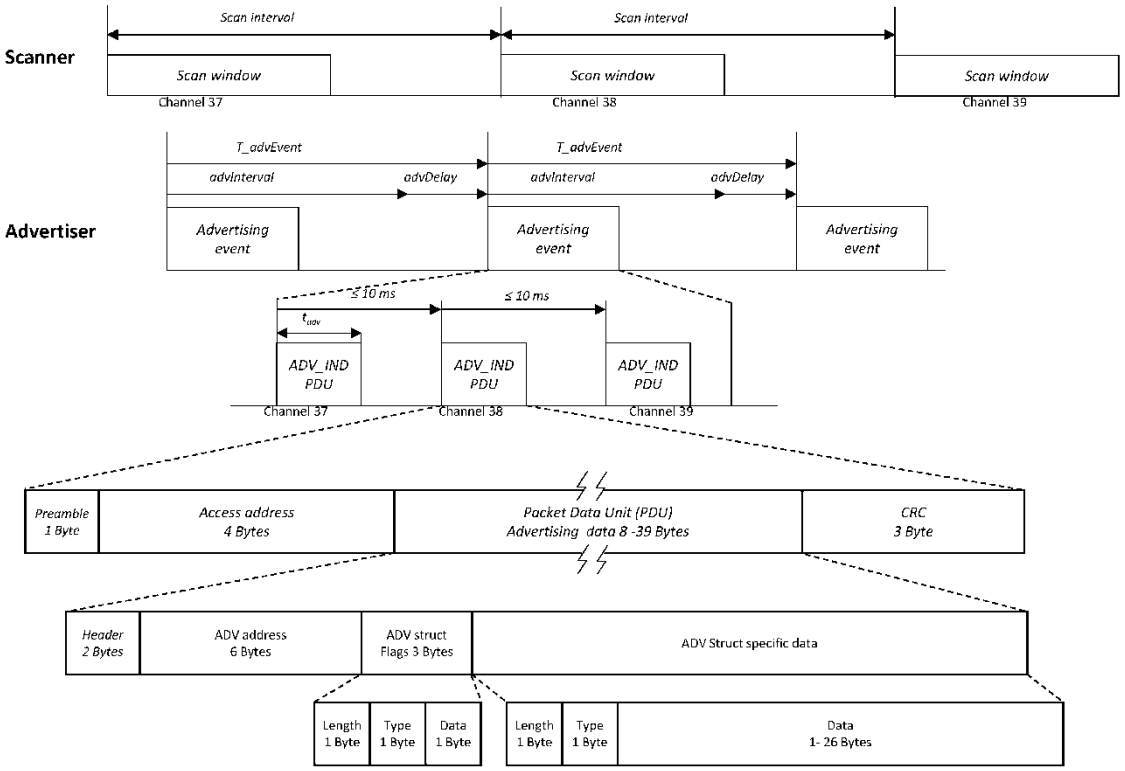

Fig 1.3 Transmisión de advertisement BLE y estructura del paquete

El paquete de advertisement o advertisement Packet Data Unit (PDU) sigue la estructura descrita en la parte inferior de la Figura 1.3.

Los dispositivos BLE pueden establecer comunicación de dos maneras: con radiodifusiones o estableciendo conexiones.

- Radiodifusión: Envío de datos unidireccionalmente a cualquier dispositivo capaz de recoger/leer los datos transmitidos.
- Conexiones: Intercambio de datos, bidireccional, entre dos dispositivos.

Los perfiles de Bluetooth definen las características y funciones necesarias de cada nivel en el sistema Bluetooth.

BLE tiene dos perfiles genéricos: GAP (Generic Access Profile) y GATT (Generic Attribute Profile).

- GAP: Es la capa de control de BLE, gestiona las radiodifusiones y conexiones en Bluetooth. Permite que un dispositivo sea visible por otro y determina como deben de interactuar dos dispositivos entre ellos.
- GATT: Es la capa de datos de BLE, una especificación general de la comunicación (transmisión y recepción) de tramas cortas.

GATT tiene dos perfiles:

- Cliente: Envía peticiones al servidor y recibe respuestas desde el servidor. El cliente GATT no conoce a priori nada sobre los atributos del servidor, así que lo primero que debe hacer es preguntar a cerca de la presencia y naturaleza de los atributos mediante el descubrimiento de servicios.

- Servidor: Recibe peticiones desde el cliente y envía respuestas. Es el responsable de almacenar los datos, organizados en atributos, y hacer posible su uso por el cliente.

Además GATT puede tener tres tipos de atributos: Característica, servicio y descriptor. La característica es el dato que se transmite entre usuario y servidor, el servicio es una colección de características que se utilizan conjuntamente para proporcionar una funcionalidad en particular. Y por último, el descriptor es una información adicional sobre las características.

1.2.1.1 PDU de advertisement packet

La PDU de un paquete advertising tiene una longitud máxima de 39 bytes, dos de los cuales corresponden a la cabecera y el resto a la carga útil (payload). La cabecera se divide en seis campos diferentes los cuales informan del tipo de PDU que es y sus características principales. En la tabla siguiente se muestra un esquema de la estructura de la PDU de un paquete advertising. [2]

Advertising PDU (hasta 39 bytes)					
Cabecera (2 bytes)					Payload (hasta 37 bytes)
PDU type (4 bits)	RFU (2 bits)	TxAdd (1 bit)	RxAdd (1 bit)	Length (6 bits)	RFU (2 bits)	

Tabla. 1.1 Trama PDU de un paquete de advertising

El campo TxAdd indica que la dirección contenida en el campo MAC address es pública (TxAdd=0) o aleatoria (TxAdd=1). El campo RxAdd indica si la dirección del iniciador del campo data es pública (RxAdd=0) o aleatoria (RxAdd=1).

El campo Length indica la longitud del payload de la PDU y el campo RFU es un campo reservado para usos futuros.

Y quizás el campo más importante es el PDU type el cual indica cuál de los siete tipos de PDU que existen para un paquete advertising se utiliza.

Hay definidos siete tipos de advertisement PDU. Cuatro pueden ser utilizadas en el estado de advertisement (ADV_*), otros cuatro en el estado de escaneo (SCAN_*) y uno en el estado de iniciación (CONNECT_REQ).

PDU type	Packet Name
0000	ADV_IND
0001	ADV_DIRECT_IND
0010	ADV_NONCONN_IND
0011	SCAN_REQ
0100	SCAN_RSP
0101	CONNECT_REQ
0110	ADV_SCAN_IND
0111-1111	Reserved

Tabla 1.2. Tipos de PDUs de advertising

Debido a que en nuestro sistema los beacons únicamente estarán en el estado de advertising, solo serán definidos los cuatro tipos relacionados con este estado.

- ADV_IND: Este tipo de PDU se conoce como conectable y no dirigido (connectable undirected). Que sea conectable indica que un dispositivo central puede conectarse a uno periférico que esté enviando este tipo de advertisings. Que no sea dirigido significa que los paquetes se envían en modo broadcast.

- ADV_DIRECT_IND: Este tipo de PDU es conectable y dirigida (connectable directed) y se utiliza cuando el advertiser quiere conectarse con un dispositivo concreto. Este tipo de paquete se utiliza para solicitar la conexión a otro dispositivo de manera directa.

- ADV_NONCONN_IND: Este tipo de PDU es no conectable y no dirigido (non-connectable undirected) y se utiliza para emitir datos en modo broadcast, sin intención de establecer ningún tipo de conexión con los dispositivos que reciban los paquetes. Los dispositivos scanner que detecten estos paquetes únicamente pueden leer los datos del paquete.

- ADV_SCAN_IND: Este tipo de PDU no dirigido pero escaneable (scannable undirected) se utiliza para emitir datos en modo broadcast sin permitir la conexión de ningún dispositivo. Sin embargo, dejan que los que los reciben puedan interactuar con ellos y solicitarles información adicional.

En este proyecto se trabajará con el estándar de Bluetooth 4 y en lo que se refiere al tipo de advertisement, se utilizará ADV_NONCONN_IND debido a que es el que más se ajusta a los requerimientos del sistema.

1.2.2. Aplicación del BLE

En el proyecto necesitamos que el emisor BLE haga radiodifusiones, es decir, que se comporte como un beacon. Las radiodifusiones envían paquetes de difusión (advertising packet). Estos paquetes son enviados cada cierto tiempo (advertising interval).

El perfil de Cliente lo realizaría el móvil del voluntario mientras que el perfil de Servidor lo realizaría el chip del dorsal.

La característica, dato que se transmite entre cliente y servidor, sería el identificador del corredor.

1.2.3. Beacons Bluetooth Low Energy

Un beacon es un pequeño transmisor de radio Bluetooth, alimentado por baterías. Estos pequeños dispositivos de hardware transmiten incesantemente señales de Bluetooth de baja energía (BLE). Los teléfonos inteligentes habilitados para Bluetooth son capaces de escanear y mostrar estas señales.

Es importante comprender la diferencia entre Bluetooth clásico y Bluetooth de bajo consumo para apreciar los beacons BLE. El Bluetooth clásico consume alta potencia debido, entre otras cosas, a su transmisión continúa de datos.

Sin embargo, Bluetooth Low Energy transmite menos datos, por lo que consume mucha menos energía. Los beacons BLE transfieren pequeñas cantidades de datos a intervalos regulares de tiempo.

Fig 1.4 Escenificación de Beacon conectado a móvil

Un beacon está diseñado con tres componentes principales: una pequeña computadora ARM, un módulo de conectividad Bluetooth Smart y baterías para alimentar todo el circuito. La CPU de la computadora ARM tiene una antena conectada que emite ondas electromagnéticas con una longitud y frecuencia específicas [3].

Los beacons transmiten paquetes de datos que, en general, solo contienen una ID con sus datos específicos o datos variables de control (la temperatura, el estado de la batería, el identificador del corredor, etc.).

Independientemente del proveedor que fabrique el beacon, todos ellos tienen bastantes similitudes en sus componentes de hardware y firmware.

1.2.3.1 Hardware

El hardware básico que debe llevar un beacon consiste en una batería y un chipset que monta un microcontrolador con un chip de radio BLE. Dependiendo de las funcionalidades que se quieran obtener, a este hardware básico se le puede añadir elementos como sensores de temperatura, humedad, acelerómetros, etc.

En el chipset de un beacon encontramos el microcontrolador y el resto de sensores que haya decidido implementar el fabricante.

La batería de un beacon puede tener una duración que oscile entre unos pocos meses o años dependiendo del tipo de batería y de parámetros configurables como la potencia de transmisión y el advertisement interval. A mayor potencia de transmisión y menor advertisement interval más aumentará el consumo de batería y por lo tanto se agotará antes.

1.2.3.2 Firmware

Cada fabricante aporta a sus beacons un firmware específico que aporta toda la lógica de bajo nivel para hacer funcionar su hardware y de esta manera poder controlar diferentes parámetros del dispositivo.

En el caso de los beacons, el firmware puede controlar dos parámetros importantes como son la potencia de transmisión y su advertisement interval de advertising. Estos parámetros, en el caso particular de nuestro modelo, puede ser programados como veremos en apartados siguientes por una plataforma de desarrollo llama Mbed.

Los beacons transmiten su señal de advertising a una potencia fija pero configurable según los requerimientos de la aplicación a la que están destinados. En el momento de configurar la potencia hay que tener en cuenta que la señal pierde potencia debido a la atenuación por distancia u otros obstáculos.

También puede verse afectada por las interferencias del entorno u otras tecnologías que trabajen en la misma frecuencia de emisión.

Típicamente los beacons emiten en un intervalo de 100ms. Los fabricantes permiten configurar estos intervalos desde cada 100ms a cada 10s, proporcionando de esta manera un rango de libertad para el programador de manera que pueda elegir el valor que más le convenga para su aplicación.

El advertisement interval elegido proporcionará una capacidad de respuesta del dispositivo receptor mayor o menor dependiendo de si el intervalo es más corto o más largo.

1.3. Comparación RFID contra BLE

La comparación entre los dos sistemas se basa en tres puntos bien diferenciados: Accesibilidad, alcance/precisión y coste. [4]

1.3.1. Accesibilidad

La implementación de un sistema RFID requiere una serie de componentes diferentes, que incluyen el hardware básico como etiquetas, lectores, control de lectores y un software de aplicación.

Además, si bien la mayoría de los sistemas de seguimiento de ubicación tienen una compatibilidad inherente con el móvil, la RFID no lo hace. RFID requiere hardware / firmware que pueda procesar señales en frecuencias específicas.

En el caso de un sistema de localización BLE, los beacons funcionan como transmisores de señales que funcionan principalmente con baterías y se pueden configurar con la ayuda de una aplicación móvil. Esto los hace escalables y altamente portátiles. Además de eso, la capacidad de los beacons para permitir que los teléfonos inteligentes actúen principalmente como receptores hace que sea una tecnología de ubicación altamente accesible. En contraposición, los beacons una vez instalados deben revisarse regularmente para ver los niveles de batería, etc.

1.3.2. Alcance y precisión

El alcance se define como la distancia que recorre la señal, depende de la configuración de energía y el entorno en el que se implementa. Los beacons suelen tener un alcance inalámbrico de 1 a 70 m.

La confiabilidad de la señal dentro de un alcance dado, y la tolerancia de esa señal cuando se tienen en cuenta los factores ambientales se conocen como precisión.

Si bien la RFID permite identificar hasta 1000 etiquetas por segundo a velocidades de lectura de casi el 100%, su precisión varía según la frecuencia: cuanta más alta es la frecuencia, más sensible es a la interferencia. Esta es una de las razones principales por las que las etiquetas UHF RFID no se utilizan para rastrear productos metálicos o aquellos con alto contenido de agua.

Debido a que los beacons son transmisores de radio, estos son propensos a sufrir interferencias, ya que las señales de radio pueden ser absorbidas por diferentes medios, como agua, aire, cuerpos humanos o incluso superficies metálicas.

También hay que tener en cuenta que el alcance de la transmisión depende de la potencia de transmisión, cuento mayor sea esta mayor será el alcance lo que desafortunadamente lleva a una descarga de batería más rápida. Es decir a mayor alcance menos duración de batería.
Dicho esto, la potencia de transmisión puede variarse según el caso de uso en cuestión.

1.3.3. Coste

El coste asociado con las soluciones RFID o beacons BLE depende del tipo de solución que se planea implementar. Por ejemplo, cuando se trata del seguimiento a nivel de unidad de inventario, RFID es, sin lugar a dudas, la solución de automatización más barata que existe. Las etiquetas pasivas pueden costar tan poco como 1 céntimo y algunos sistemas de lectores de frecuencia UHF pueden procesar 1.000 etiquetas por segundo.

Por otro lado, si el seguimiento es de activos a nivel de objeto grande (por ejemplo, equipos o contenedores de carga), los beacons podrían ser más rentables en comparación con los sistemas RFID. Además, aunque el coste de una etiqueta RFID activa está a la par con el de un beacon, requiere una inversión más costosa en el lado de la infraestructura del lector. Un solo lector puede costar hasta 1,500€, mientras que casi todos los Smartphones en el mercado pueden actuar como lectores en el caso de las beacons.

CAPÍTULO 2. IMPLEMENTACIÓN SISTEMA BEACONS BLE

2.1. Funcionamiento del sistema

El proyecto consiste en la implementación de un sistema de seguimiento de los corredores de las carreras populares.

El sistema funciona bajo tres puntos básicos: la tecnología BLE, el servidor y la aplicación móvil.

El funcionamiento del sistema parte de dos estados iniciales:

- El corredor lleva en el dorsal un beacon BLE que va transmitiendo su identificador.

- El voluntario tiene en su Smartphone (compatible con la tecnología BLE) la aplicación para recoger los datos desde los beacons de cada corredor.

En el momento en el que el corredor entra en el rango de cobertura BLE del móvil, este detecta el identificador del corredor que está enviando el beacon del dorsal. Una vez la aplicación obtiene el identificador del corredor, esta procesa los datos de tiempo y posición GPS, la cual se obtiene del mismo móvil. Cuando se ha procesado toda la información, la aplicación la sube al servidor para que este la gestione.

El servidor tiene dos funciones:

- Recibir los datos desde la aplicación y guardarlos en la base de datos.

- Mostrar a través de una interfaz web, los datos recogidos durante las carreras junto con estadísticas de la misma.

2.1.1. ¿En qué mejora este al sistema con RFID?

La mejora más importante que aporta nuestro sistema con BLE sobre el habitual basado en RFID es la posibilidad de poder obtener un dato, tanto de tiempo como de posición GPS, por cada voluntario que haya en la carrera en vez de tener que instalar lectores RFID a lo largo de todo el recorrido.

Además el hecho de no necesitar lectores específicos, como sí son necesarios con el sistema RFID, hace que la inversión en la infraestructura de la carrera sea más barata debido a que se utiliza como lector de datos los Smartphones de los voluntarios.

2.2. Dispositivo BLE

Para desarrollar el sistema, el departamento de TSC de la universidad me ha proporcionado un dispositivo BLE para utilizarlo como Beacon. Este dispositivo simulará el dispositivo BLE que llevarían los corredores en el dorsal.

2.2.1. Modelo

El dispositivo BLE que me han proporcionado es el modelo ReadBearLab Nano [5]. BLE Nano es la placa de desarrollo de Bluetooth 4.1 Low Energy (BLE) más pequeña del mercado. El núcleo es Nordic nRF51822 (con un microcontrolador ARM Cortex-M0 SoC más BLE) que funciona a 16MHz con un consumo de energía ultra bajo. BLE Nano funciona a menos de 1,8 V o 3,3 V, por lo tanto es compatible con muchos componentes electrónicos.

Figura 2.1 Dispositivo BLE ReadBearLab Nano

Este dispositivo debe de ser programado de tal manera que sea útil para la función que debe cumplir en el sistema. La programación se hace a partir de la plataforma Mbed.

2.2.2. Mbed

Mbed es una plataforma y sistema operativo para dispositivos conectados a Internet basados en microcontroladores ARM Cortex-M de 32 bits. Tales dispositivos también se conocen como dispositivos de IoT. El proyecto es desarrollado en colaboración por Arm y sus socios tecnológicos. [6]

Las aplicaciones para la plataforma Mbed se pueden desarrollar utilizando el IDE en línea de Mbed, un editor y compilador de código en línea gratuito. Solo es necesario instalar un navegador web en la PC local, ya que un proyecto se compila en la nube, es decir, en un servidor remoto, utilizando el compilador ARMCC C / C ++.

El IDE de Mbed proporciona espacios de trabajo privados con la capacidad de importar, exportar y compartir código con el control de versión de Mercurial distribuido, y puede usarse también para la generación de documentación de código.

Las aplicaciones también se pueden desarrollar con otros entornos de desarrollo como Keil µVision, IAR Embedded Workbench y Eclipse con herramientas integradas GCC ARM.

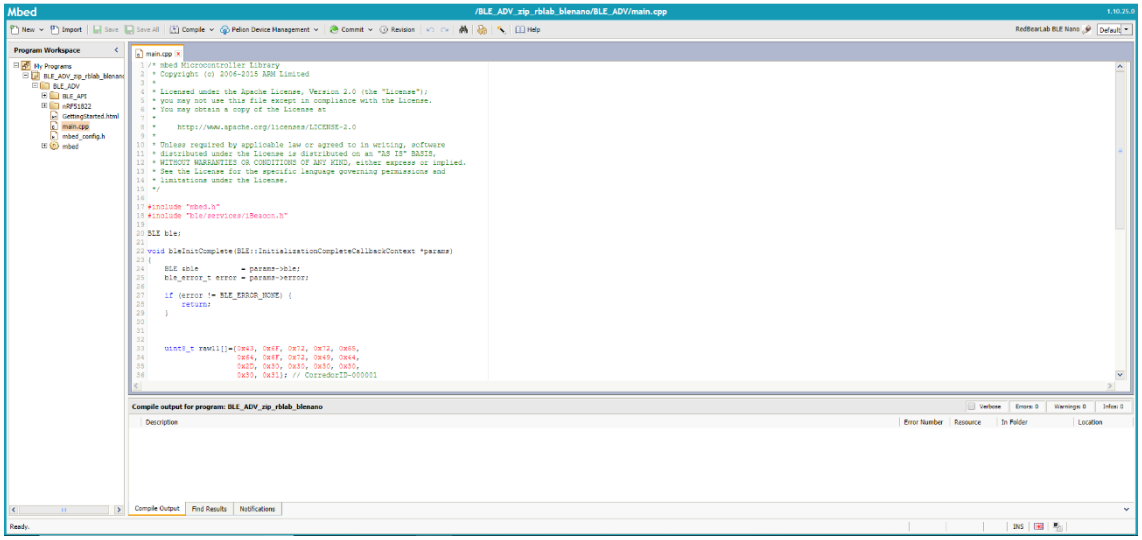

Figura 2.2 Workspace de Mbed

2.2.3. Proceso de programación del dispositivo

El primer paso del proceso de programación del dispositivo BLE es crear un Workspace para el dispositivo en el que compilaremos el programa. Una vez elegido el dispositivo y creado el Workspace, el siguiente paso es importar una plantilla básica proporcionada por el departamento TSC en la cual se indican los parámetros y funciones básicas para que el programa funcione correctamente.

En el caso de nuestro sistema, el único requerimiento que tenemos es el nombre del dispositivo, el cual como veremos en apartados siguientes debe seguir la estructura siguiente: CorredorID-XXXXXX. Otros parámetros como el advertisement interval o la potencia de transmisión también se configuran pero en este caso los dejamos en valores por defecto.

A continuación se muestra un ejemplo en el que se crea un array para programar el nombre CorredorID-000001:

```
uint8_t raw11[]={0x43, 0x6F, 0x72, 0x72, 0x65,
                 0x64, 0x6F, 0x72, 0x49, 0x44,
                 0x2D, 0x30, 0x30, 0x30, 0x30,
                 0x30, 0x31};
```

Como se puede observar el array tiene 17 elementos, que son el mismo número de caracteres que tiene el nombre del dispositivo. Cada elemento del array es equivalente a cada carácter del nombre en hexadecimal.

Una vez el código está listo, este se compila y si en el caso de que este correcto, automáticamente se descargara un archivo con formato .hex.

Por otro lado, cuando se conecta el dispositivo al puerto USB del ordenador, automáticamente se abre una unidad, como si fuese una memoria USB, que en el caso del modelo ReadBearLab Nano se llama DAPLINK. Es en esta unidad donde hay que copiar el fichero .hex y es el propio software del ReadBearLab quien instalará el programa compilado.

Una vez instalado el programa en el dispositivo y después de reiniciarlo, este dispositivo ya emitirá advertisements con el nombre CorredorID-000001. La Figura 2.3 es una captura de pantalla de la aplicación de móvil nRF Connect, la cual se utiliza para escanear dispositivos Bluetooth. En esta captura se puede ver como el nombre del dispositivo es el correcto y además el valor del campo de tipo 0x09 coincide perfectamente con los valores del array que hemos visto anteriormente.

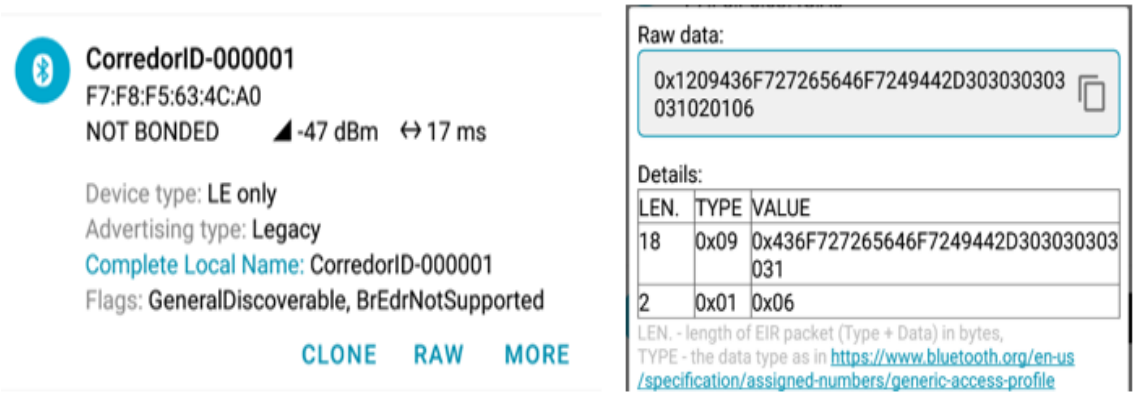

Figura 1.3 Información y propiedades del Beacon programado

En el apartado 6.3 del Anexo, se adjunta todo el código utilizado para programar el dispositivo.

2.3. Servidor

El proyecto se desarrolla a nivel local, por lo tanto necesitamos de un localhost que nos facilite la creación de un servidor web y un sistema de gestión de bases de datos. Después de una búsqueda exhaustiva XAMPP ha sido el software decidido para llevarlo a cabo.

La decisión de utilizar el software XAMPP tiene dos motivos principales: su previo uso en la asignatura PES (Projecte d'enginyeria del software) y la recomendación de su uso para localhost en foros importantes como StackOverflow.

2.3.1. XAMPP

XAMPP es un paquete de software libre, que consiste principalmente en el sistema de gestión de base de datos MySQL (MariaDB), el servidor web Apache y los intérpretes para lenguajes de script PHP y Perl [7].

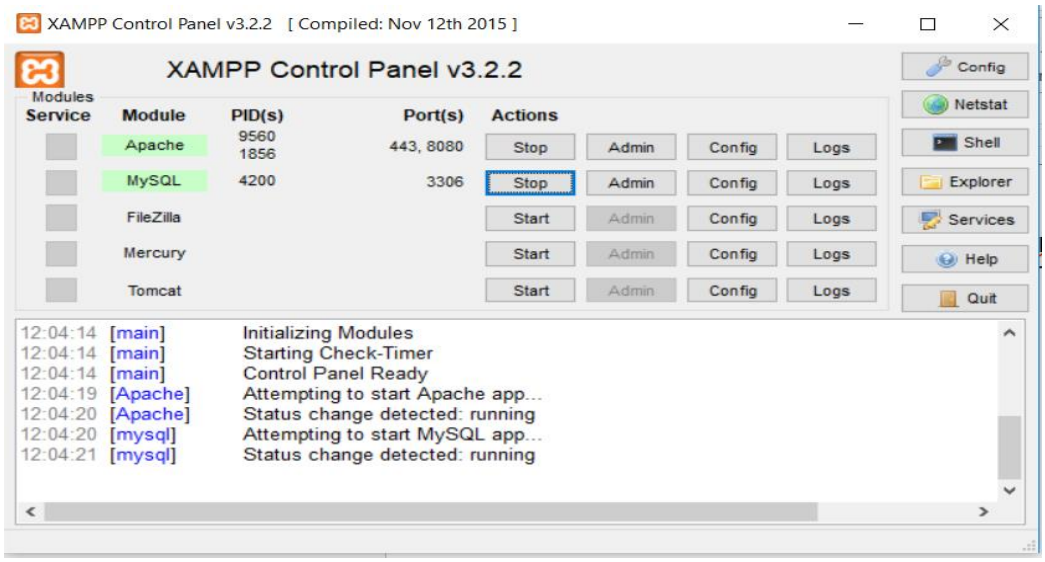

Fig 2.4 Panel de control de XAMPP

En la captura anterior, podemos observar el panel de control de XAMPP donde se ve como está activo el módulo Apache en el puerto 8080 y el módulo MySQL en el puerto 3306. El módulo Apache es el que da soporte al servidor web y el MySQL a la base de datos.

Para poder acceder al servidor web desde el navegador deberemos poner la dirección IP del pc donde este alojado el servidor junto con el puerto. Ejemplo:

- Sí se accede desde el mismo PC que el servidor:
  http://localhost:8080/RoT.html

- Sí se accede desde otro PC de la misma red local:
  http://192.168.1.42:8080/RoT.html

Cualquier comunicación entre el servidor y la base de datos se hace a partir de Hypertext Preprocessor (PHP).

PHP es un lenguaje de programación del lado del servidor cuyo código es interpretado por un servidor web con un módulo de procesador PHP que genera un HTML resultante. Una gran ventaja de PHP es que puede combinarse con MySQL para trabajar con bases de datos.

2.3.2. Base de datos

La base de datos que proporciona XAMPP es del sistema MariaDB, el cual es una evolución de MySQL y en la cual se incorporan mejoras con más funcionalidades y un mayor rendimiento tanto a nivel de velocidad como de consumo de recursos.

XAMPP desde su panel de control ofrece la herramienta PhpMyAdmin con la que podremos administrar la base de datos de una manera mucho más visual e intuitiva.

2.3.2.1 Estructura de la base de datos

La base de datos está formada por 6 tablas: voluntarios, corredores, carreras, datos, carreras_voluntarios y carreras_corredores. Estas dos últimos serían necesarias para poder tener las relaciones m:n entre voluntarios, carreras y corredores.

En la Figura 2.5 que se muestra a continuación se puede ver una representación de la base de datos y sus relaciones.

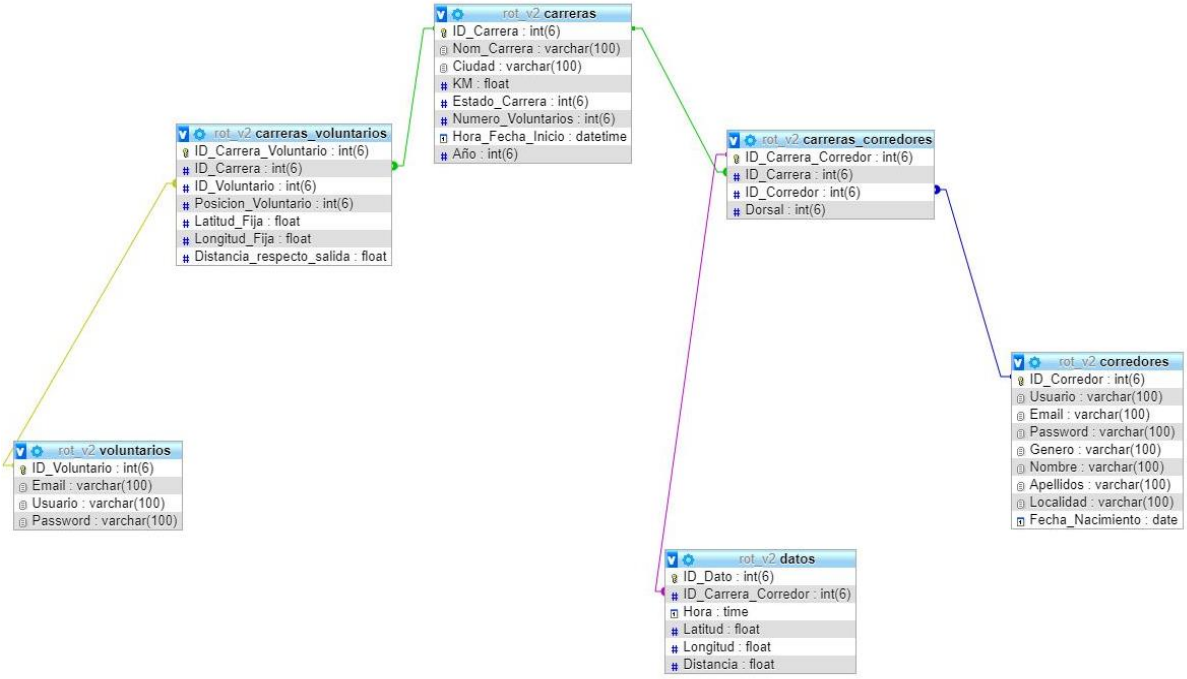

Fig 2.5 Diseño de la base de datos

En la base de datos existen las siguientes relaciones:

- Voluntarios y carreras: La relación entre estas dos tablas es m:n ya que una carrera tiene muchos voluntarios y un mismo voluntario puede serlo en diferentes carreras.
  Esta relación m:n hace necesaria la creación de una tabla auxiliar carreras_voluntarios.

- Corredores y carreras: La relación entre estas dos tablas es m:n ya que una carrera tiene muchos corredores y un corredor puede participar en diferentes carreras.
  Igual que en la anterior, el hecho de que sea una relación m:n hace necesaria la creación de una tabla auxiliar carreras_corredores.

- Carreras_corredores y datos: La relación entre estas dos tablas es 1:m ya que la idea de la tabla datos es guardar todos los datos del corredor captados por los voluntarios durante la carrera, es decir un corredor tiene muchos datos de una misma carrera, es por eso que la relación es con la tabla carreras_corredores.

Como se puede ver en la figura anterior, cada dato tiene un campo de latitud y otro de longitud. Además, en la tabla Carreras_voluntarios existen los campos latitud_fija y longitud_fija. Estas coordenadas fijas solo se utilizan en el caso de que la señal GPS del dispositivo móvil sea débil o inexistente. Dichas coordenadas fijas son puestas por primera vez por el administrador de la carrera, pero una vez el voluntario correspondiente envíe un dato con coordenadas GPS, estas serán actualizados de manera que la próxima vez que se requieran estas coordenadas fijas sean más precisas que las impuestas por el administrador de la carrera.

2.3.3. Web

Como se ha explicado en la introducción del proyecto, una de las bases es poder consultar los datos de las carreras a partir de una interfaz web.

La web se estructura bajo tres grandes pilares: consultas generales, sección corredores y sección voluntarios.

2.3.3.1 Página principal

Desde la página principal de la web se puede acceder a dos tipos de consultas, recorrido de una carrera y tiempos de un corredor en una carrera, además de poder acceder a la página de inicio de sesión o a la de información de la web “sobre nosotros”.

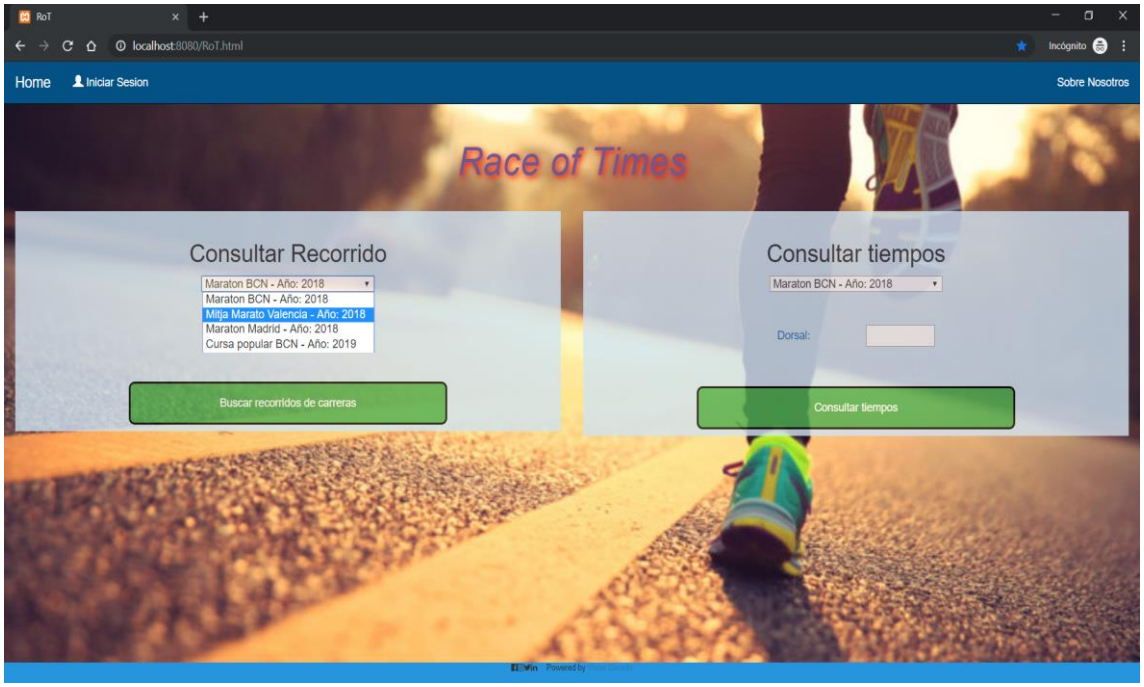

Fig 2.6 Página principal de la web

Como se puede ver en la Figura 2.6, se pueden hacer dos tipos de consultas. La primera es una consulta sobre el recorrido de una cierta carrera escogida en el desplegable donde aparecen todas las carreras existentes y la segunda es consultar los tiempos que ha obtenido un corredor en una carrera a partir de proporcionar el dorsal que llevaba ese corredor en dicha carrera.

En el caso de que en la consulta de tiempos se seleccione una carrera que todavía no ha sido iniciada, aparecerá un aviso de que esa carrera no ha empezado y por lo tanto no tiene tiempos.

2.3.3.2 Página inicio sesión

Como se ha comentado anteriormente, dos pilares básicos de la web son las secciones de voluntarios y corredores. A cada una de estas secciones solo se puede acceder a través de un inicio de sesión.

Como podemos ver en la Figura 6.1 situada en el anexo, el inicio de sesión consta de dos campos donde poner el usuario y contraseña. Además contiene dos selectores donde elegir sí se quiere iniciar sesión como corredor o voluntario.

2.3.3.3 Página corredor

Una vez realizado el inicio de sesión como un corredor llegamos a esta página dedicada al corredor.

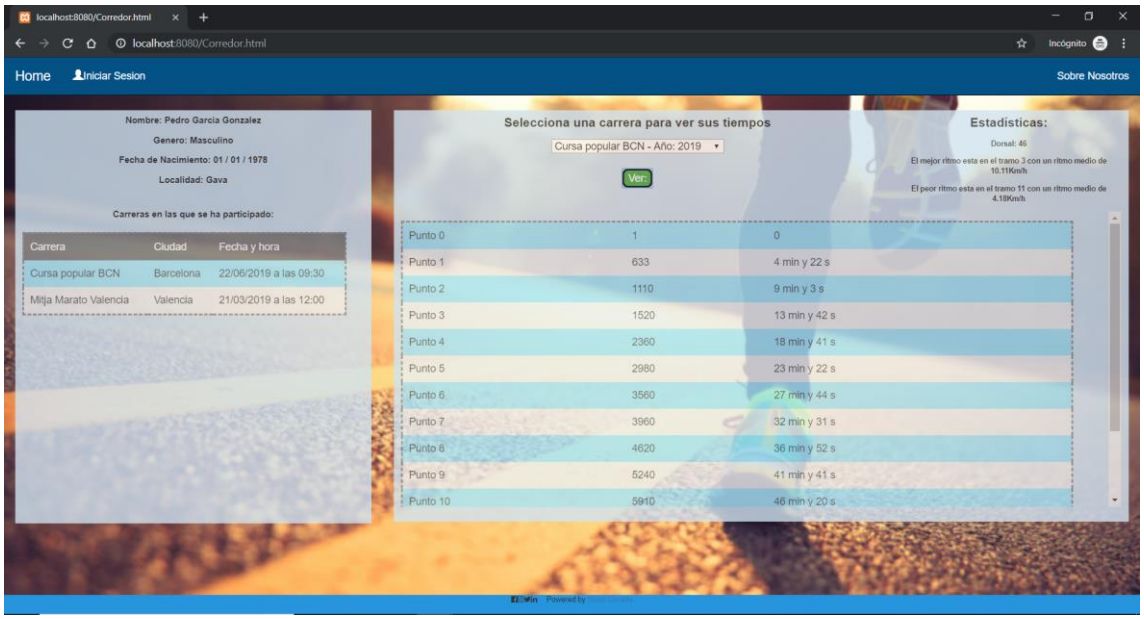

Fig 2.7 Página dedicada al corredor

Como se puede ver en la Figura 2.7, el diseño de la página está claramente separado en dos zonas: La zona izquierda, donde se puede ver los datos personales del corredor tales como nombre, género, fecha de nacimiento y su localidad. Además también se muestra una tabla de todas las carreras en las que ha participado ese corredor. En la zona derecha, podemos ver como hay un selector de carreras con el cual podemos seleccionar la carrera de la cual queremos ver sus datos (distancia y tiempo transcurrido entre cada punto de control) más en profundidad o estadísticas de los tramos con mayor y menor ritmo.

2.3.3.4 Página voluntario

En el caso que realicemos el inicio de sesión como un voluntario llegamos a la página dedicada al voluntario.

Como podemos ver en la Figura 6.2 del anexo, la página dedicada al voluntario solo consta de dos tablas prácticamente iguales donde se muestra especificaciones de cada carrera tales como el nombre, la fecha en la que se llevará a cabo y la ciudad donde se realizará. La diferencia entre cada tabla es que en la de la izquierda se muestran las carreras ya finalizadas mientras que

en la tabla situada a la derecha se muestra las carreras en curso o las que todavía no se han iniciado.

2.3.3.5 Página recorridos

Esta página es una de las dos páginas principales a la que se accede desde la página principal seleccionando la carrera de la cual se quiere ver el recorrido.

Como se puede ver en la captura de pantalla Figura 6.3 del Anexo, la parte importante de la interfaz es el mapa, donde se puede ver la localización de cada voluntario gracias a un marcador además del recorrido de la carrera marcado por la línea azul.

En la zona izquierda se encuentra un apartado donde a partir de una elección de punto de carreras se muestra información sobre él, como por ejemplo la distancia recorrida en la carrera hasta ese punto de control.

2.3.3.6 Página tiempos de carreras

Esta página se puede considerar la más importante de toda la web, ya que es la que muestra el objetivo principal del proyecto. Se accede a partir de la página principal indicando la carrera y el dorsal del corredor del cuál quieres ver los datos y estadísticas.

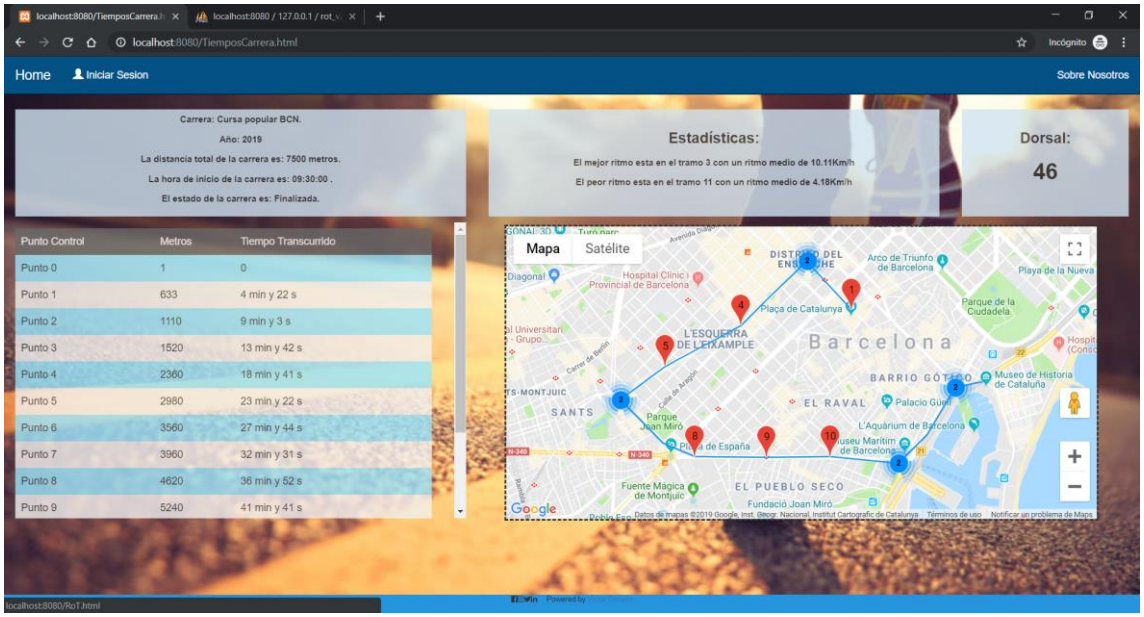

Fig 2.8 Página tiempos de carrera

Como se puede observar la interfaz de esta página se puede dividir en tres zonas claramente diferenciadas:

La parte superior donde se describe la carrera (nombre, fecha y hora de inicio y estado de la carrera), se indica el dorsal que se ha consultado y además se muestran estadísticas de mejor y peor ritmo durante la carrera.

En la parte izquierda se muestra una tabla con la distancia recorrida y el tiempo transcurrido en cada punto de control donde recordamos se recogen estos datos a partir del voluntario.

Por último, en la parte derecha hay un mapa, que igual que en la página de recorridos de carrera, muestra los puntos de control y el recorrido seguido por el corredor.

2.3.4. Retos técnicos de la programación

El diseño de todo el sistema del servidor-web, ha conllevado varios retos importantes: la conexión entre Frontend y Backend, la implementación de los mapas con la API de google y el traspaso de información entre páginas.

2.3.4.1 Conexión Frontend - Backend

El frontend son todas aquellas tecnologías que corren del lado del cliente, es decir, todas aquellas tecnologías que corren del lado del navegador web, generalizándose más que nada en tres lenguajes, HTML, CSS y JavaScript.

Por otro lado, tenemos el Backend enfocado en hacer que todo lo que está detrás de un sitio web funcione correctamente. Toma los datos, los procesa y los envía al usuario, además de encargarse de las consultas o peticiones a la Base de Datos, la conexión con el servidor, entre otras tareas. Cuenta con una serie de lenguajes y herramientas que le ayudan a cumplir con su trabajo como PHP, Ruby, Python, JavaScript, SQL, MongoDb, MySQL, etc.

Fig 2.9 Funcionamiento web: Backend y Frontend

En la Figura anterior se puede observar un esquema del funcionamiento de la web donde a la parte izquierda quedaría el Frontend y a la derecha el Backend.

En el caso de nuestro proyecto en el que utilizamos la herramienta XAMPP nos facilita la gestión de la base de datos a nivel interno como podemos ver en el apartado 2.3.2.

Para poder hacer la conexión con la base de datos en MySQL es necesario utilizar PHP. El procedimiento sería el siguiente: desde HTML se solicita una consulta que está en un fichero PHP externo en el servidor, el servidor ejecuta está script PHP que contiene la consulta MySQL para obtener los datos solicitados de la base de datos y devuelve los datos a HTML para que mediante JavaScript los gestione de la manera que necesite. De esta manera se consigue una web más dinámica.

A continuación se muestra una parte del código HTML de la página Inicio de sesión, en la cual se hace la consulta al servidor para hacer el inicio de sesión.

```
var usu= document.getElementById("uname").value;
var pswt= document.getElementById("psw").value;
var tipot=document.querySelector('input[name="tipousuario"]:checked').value;
var hr = new XMLHttpRequest();
var url = "check-login.php";
var vars = "uname="+usu+"&psw="+pswt+"&tipousuario="+tipot;
hr.open("POST", url, true);
hr.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
hr.onreadystatechange = function() {
  if(hr.readyState == 4 && hr.status == 200){
    var return_data = hr.responseText;
    traspaso(return_data);
  }
}
hr.send(vars);
```

Fig 2.10 Código conexión inicio de sesión en HTML

Para establecer la conexión entre el HTML y el servidor se utiliza la librería XMLHttpRequest. Como se puede ver en la Figura 2.10 esta librería permite la conexión con el fichero PHP encargado de verificar el inicio de sesión en el servidor.

Para poder enviar todos los datos necesarios para la comprobación con una seguridad mínima se utiliza el método Post, de esta manera el valor de las variables usuario y contraseña no puede ser visto en la URL de navegación del navegador de internet.

Como se puede ver en la Figura 2.10, el propio objeto crea un callback (onreadystatechange) el cual será llamado en el momento en el que el servidor responda. Y será la variable hr.responseText quién contenga la respuesta del servidor de tipo String.

2.3.4.2 Traspaso de información entre páginas

Uno de los problemas más importantes que surgió durante el desarrollo de la página web es el traspaso de información entre páginas. Por ejemplo, si en la página principal se busca unos tiempos de un dorsal en una cierta carrera, cómo pasar el valor del dorsal y el nombre de la carrera a la página que se direcciona par que esta pueda consultar la información necesaria a la base de datos.

La solución aplicada es el uso de la propiedad sessionStorage. La propiedad sessionStorage permite acceder a un objeto Storage asociado a la sesión actual.

La información almacenada en sessionStorage es eliminada al finalizar la sesión de la página. La sesión de la página perdura mientras el navegador se encuentra abierto, y se mantiene por encima de las recargas y reaperturas de la página.

En la Figura 2.11 se muestra como se almacenan variables en el sessionStorage con el comando sessionStorage.setItem(). Además la última sentencia de la figura muestra el comando location.href el cual produce que se ejecute la página que se indica.

```
sessionStorage.setItem('nombreCarrera',nom);
sessionStorage.setItem('anyoCarrera',anyo);
sessionStorage.setItem('dorsal',dorsal);
location.href ="TiemposCarrera.html";
```

Fig 2.11 Código almacenar sessionStorage

Una vez estamos en la nueva página, para cargar los datos almacenados en el sessionStorage, utilizamos el comando sessionStorage.getItem() como se puede ver en la Figura 2.12.

```
var ncarrera=sessionStorage.getItem('nombreCarrera');
var ycarrera=sessionStorage.getItem('anyoCarrera')
var dorsal=sessionStorage.getItem('dorsal')
```

Fig 2.12 Código cargar sessionStorage

2.3.4.3 Implementación de mapas

Para poder implementar la vista de los puntos de control en el mapa como se ve en la Figura 2.8, se ha utilizado la herramienta de Google: Google Cloud Platform. En esta herramienta se pone a disposición de los desarrolladores diferentes APIs de Google [8].

En este proyecto se ha utilizado la API: Maps JavaScript API, la cual permite agregar a la web un mapa, proporcionando imágenes y datos locales de la misma fuente que Google Maps. Permite editar el estilo del mapa para adaptarse a las necesidades de la web. Además permite la introducción de marcadores dentro del mapa en las coordenadas que se necesiten.

El procedimiento para implementar esta API en la web es sencillo, una vez te registras en la plataforma, creas un proyecto en el que añades las APIs que necesites y las restricciones de dominio que quieras.

Una vez realizado estos pasos, la plataforma te acredita con una clave que debes colocar en una script dentro de HTML para que pueda cargar la web.

```
<script async defer
src="https://maps.googleapis.com/maps/api/js?key=AIzaSyBBXPygEUg1RFsmNScimErwQ6C4985ByKI&callback=initMap">
</script><!-- Aquí es donde se utiliza la clave de API de google para poder utilizar lo de google maps.-->
<script src="https://developers.google.com/maps/documentation/javascript/examples/markerclusterer/markerclusterer.js"></script>
```

Fig 2.13 Scripts Api Google Maps

En la Figura 2.13 se pueden ver las dos scripts necesarias para representar los mapas en la web. La primera es la encargada de validar la API de google mientras que la segunda es la encargada de poder utilizar los marcadores de google en el mapa.

Una vez insertada la script con la credencial de la API, ya se puede personalizar el mapa de la forma que se necesite. En el caso de este proyecto poniendo marcadores en las coordenadas que se han recogido por un voluntario cuando ha enviado un dato de tiempo de un corredor. Además de pintar una línea que una cada marcador con su sucesivo.

En la siguiente figura se muestra el código que se ha utilizado para poder mostrar el recorrido en el mapa.

```
var locations=[];
  var datosPuntos=puntos[i].split("-"); //Posición, Latitud, Longitud, distancia
  //Pasamos las localizaciones a un formato que google maps reconozca
  var myLatlng = new google.maps.LatLng(parseFloat(datosPuntos[1]),parseFloat(datosPuntos[2]));
  locations.push(myLatlng);
}
var labels = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'; //Etiquetas que irán en cada localizador
//Creamos el mapa con el sitio del centro y el zoom
var map = new google.maps.Map(document.getElementById('map'), {
  zoom: 10,
  center: new google.maps.LatLng(parseFloat(datosPuntos[1]),parseFloat(datosPuntos[2])),
});
// Añadimos marcadores al mapa.
var markers = locations.map(function(location, i) {
  return new google.maps.Marker({
    position: location,
    label: ""+(i+1),
  });
});
var tripPath = new google.maps.Polyline({//Creamos la línea entre puntos
  path: locations,
  geodesic: true,
  strokeColor: '#2098F0',
  strokeOpacity: 1.0,
  strokeWeight: 2
});
tripPath.setMap(map);
// Añadimos marcadores de cluster
var markerCluster = new MarkerClusterer(map, markers,
    {imagePath: 'https://developers.google.com/maps/documentation/javascript/examples/markerclusterer/m'});
```

Fig 2.14 Código para implementar mapa

La implementación del mapa se puede dividir en cuatro fases distintas:

1) Creación de un array de coordenadas en el formato requerido por google. Esto se hace en la primera parte del código, en la cual una vez tenemos las coordenadas de los puntos, descargadas desde el servidor en una consulta previa, las procesamos con el parseador proporcionado por google: google.maps.LatLng(). Todas estas coordenadas se van guardando en un vector llamado locations.

2) Creación del mapa básico proporcionando una coordenada en la que centrarlo y un nivel de zoom. Como coordenadas para centrar el mapa se introducen la latitud y longitud del primer dato.

3) Adición en el mapa de marcadores por cada punto de coordenadas guardadas en el vector locations.

4) Dibujar una línea entre puntos siguiendo el camino (path) del vector locations.

Además de todos los pasos comentados anteriormente, en las dos últimas líneas de la Figura 2.14 se realiza una funcionalidad extra que consiste en que dependiendo del zoom del mapa, los marcadores que estén separados por poca distancia se aglomerarán en un solo marcador de color amarillo con un indicador del número de marcadores que representa. En la Figura 2.15 se muestra un ejemplo de esta funcionalidad extra.

Fig 2.15 Ejemplo aglomeración marcadores

2.3.5. Diseño responsivo

En los tiempos en que vivimos, no se puede reducir el espectro de dispositivos de acceso a web a solo ordenadores, es por eso que una característica imprescindible de una web es que sea responsiva a cualquier tipo de pantalla, tanto de ordenador como de móvil, Tablet o Smart TV.

La herramienta que hemos utilizado para crear un diseño responsivo es la librería Bootstrap 4. [9]

Bootstrap es una biblioteca multiplataforma o conjunto de herramientas de código abierto para diseño de sitios y aplicaciones web dedicada únicamente al desarrollo Frontend.

2.3.5.1 Como funciona Bootstrap

El sistema de celdas de Bootstrap está construido con flexbox y permite hasta 12 columnas en toda la página.

Si no se desea utilizar las 12 columnas individualmente, se pueden agrupar las columnas para crear columnas más anchas:

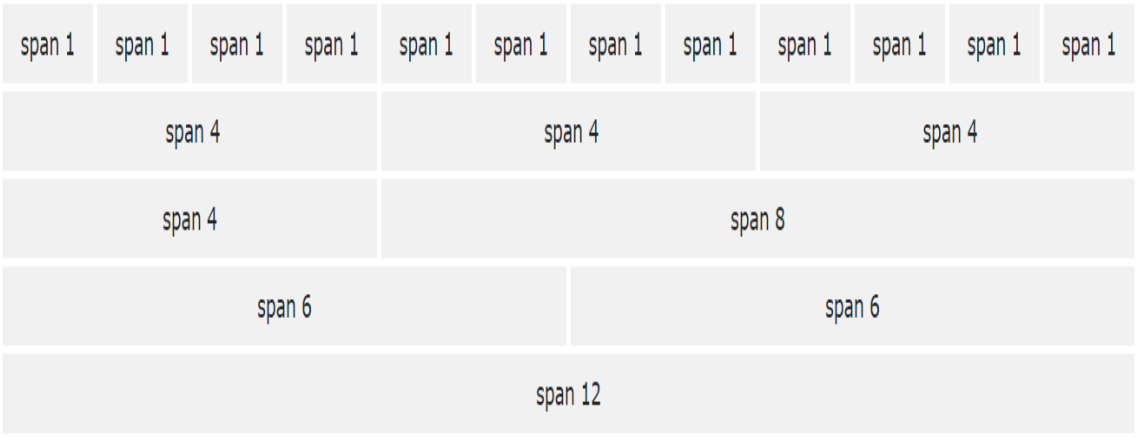

Fig 2.16 Sistema de celdas de Bootstrap

El sistema de celdas es responsivo y las columnas se reorganizarán automáticamente según el tamaño de la pantalla. Se debe asegurar que la suma de spans suma hasta 12 o menos.

El sistema de celdas de Bootstrap 4 tiene cinco clases:

- .col-xs- (dispositivos pequeños adicionales - ancho de pantalla inferior a 576px).
- .col-sm- (dispositivos pequeños - ancho de pantalla igual o superior a 576px).
- .col-md- (dispositivos medianos - ancho de pantalla igual o superior a 768px).
- .col-lg- (dispositivos grandes - ancho de pantalla igual o mayor que 992px).
- .col-xl- (dispositivos xlarge - ancho de pantalla igual o mayor que 1200px).

Las clases anteriores se pueden combinar para crear diseños más dinámicos y flexibles.

El efecto responsivo se muestra ante las clases inferiores, es decir, si se propone un diseño de dos columnas de la clase .col-md cada una con 6 de span, para las pantallas que pertenezcan a clases superiores se quedará igual, es decir, dos columnas del mismo tamaño cada una. Sin embargo en las pantallas de clases inferiores, las columnas se apilarán automáticamente una encima de la otra.

A continuación se muestra una captura de pantalla de la página principal (Figura 2.6) respondiendo a la pantalla de un móvil:

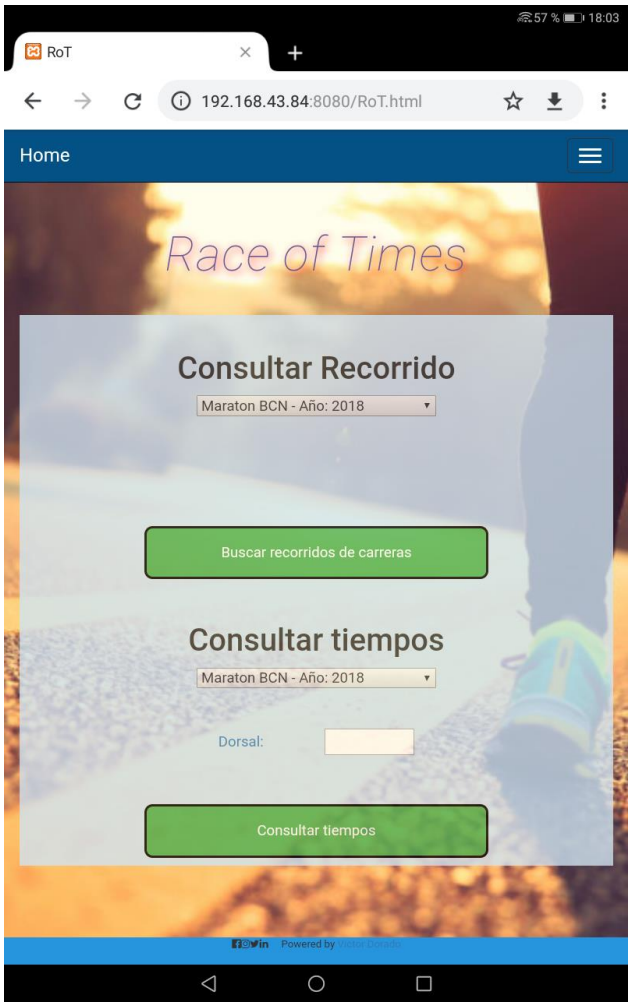

Fig 2.17 Página principal responsiva

En el caso de esta página se ha utilizado dos columnas con la clase: "col-lg-6". De esta forma en todos los dispositivos con un ancho de pantalla menor a 992 pixeles, las dos columnas se apilarán automáticamente una encima de la otra como se puede ver en la Figura 2.17.

2.4. Aplicación Android

La aplicación para el Smartphone es la tercera piedra angular junto con el servidor y el dispositivo BLE. La función de la aplicación es básicamente detectar el beacon que lleva un corredor en el dorsal y enviar a la base de datos el tiempo y coordenadas en que se ha detectado.

La aplicación se ha desarrollado con la herramienta Android Studio que es el entorno de desarrollo integrado oficial para la plataforma Android. [10]

El uso de la aplicación está restringido a los voluntarios, es decir, solo podrán iniciar sesión y hacer uso de ella los voluntarios dados de alta por el administrador del sistema.

2.4.1. Estructuración de la aplicación

La aplicación se divide en tres activitys: Inicio de sesión, menú de carreras y gestión de datos.

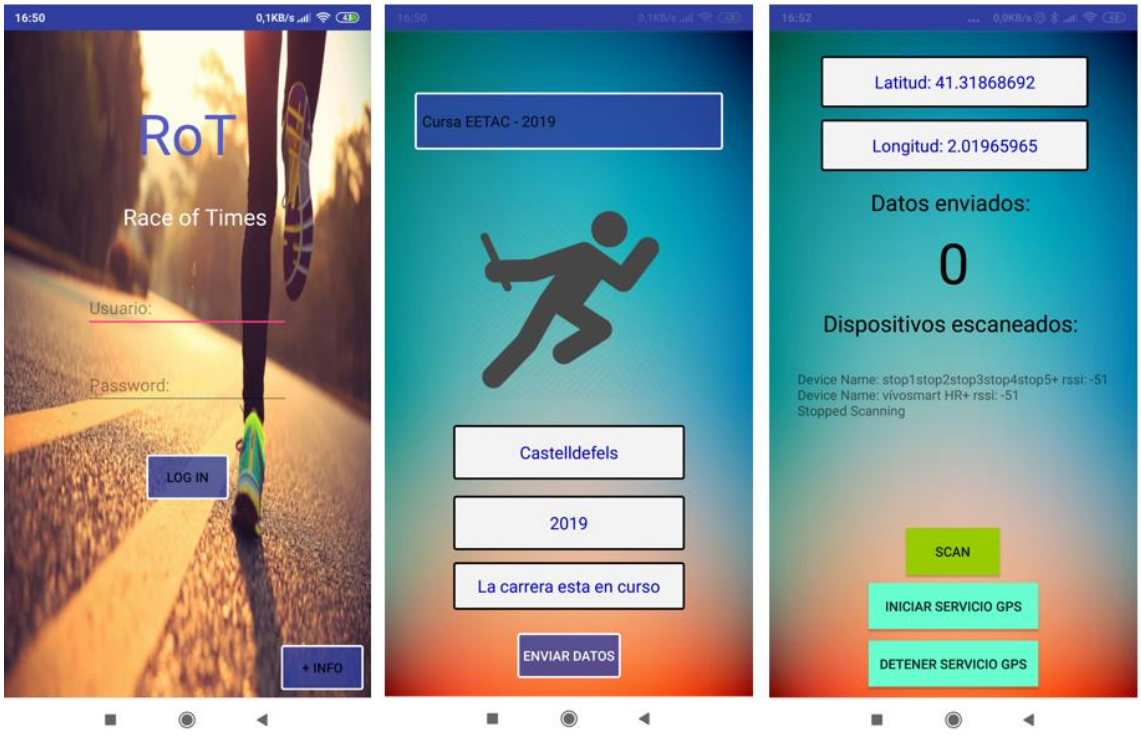

Fig 2.18 Izquierda: Inicio sesión, Central: Menú de carreras, Derecha: Gestión de datos

2.4.1.1. Activity: Inicio de sesión

La activity de Inicio de sesión es la activity inicial de la aplicación en la cual se hace el inicio de sesión para poder acceder al menú de carreras de cada voluntario.

Como se puede ver en la Figura 2.18 (Izquierda), la activity consta de dos textbox para introducir usuario y contraseña de voluntario y un botón para realizar el inicio de sesión. Además en la zona inferior derecha, hay un botón “+info” él cual si es pulsado, aparece un mensaje en pantalla de información de la aplicación con el siguiente mensaje: “Esta aplicación forma parte del TFG de Víctor Dorado. Alumno de EETAC – UPC”.

2.4.1.2. Activity: Menú de carreras

Al activity Menú de carreras se accede una vez hecho correctamente el inicio de sesión. Se recibe como parámetros desde la activity anterior el usuario con el que se ha iniciado sesión, para de esta manera, poder cargar todas las carreras en las que este voluntario ha participado.

Como se puede ver en la Figura 2.18 (Centro), en la parte superior de la activity existe un selector en el que aparecerán, una vez cargadas, todas las carreras en las que ha participado el voluntario. Una vez se haya seleccionado una de las carreras, se mostrará en los textviews de color blanco situados en la parte inferior de la activity las características principales de la carrera seleccionada: la ciudad donde se realiza, el año en el que se realiza y el estado actual de la carrera (sin iniciar, en curso o finalizada).

Por último, en la parte más inferior de la activity, existe un botón el cual permite acceder a la activity de gestión de datos, solo en el caso de que la carrera que se ha seleccionado esté en curso. Ya que en cualquiera de los otros dos estados, no tiene sentido poder acceder a gestionar datos de dorsales debido a que todavía no ha empezado la carrera o ya ha finalizado.

2.4.1.3. Activity: Gestión de datos

La activity Gestión de datos es la activity principal y más importante de toda la aplicación. Ya que es en ella en la que se escanean los beacons BLE de los dorsales de los corredores y envía los datos al servidor.

Como se puede ver en la Figura 2.18 (Derecha), la activity se puede dividir en dos zonas bastante diferenciadas: la zona superior donde se muestran las coordenadas actuales de la posición, el número de datos enviados al servidor y una lista de todos los dispositivos escaneados. En la zona inferior se puede ver tres botones donde los dos más inferiores son para iniciar y parar el servicio GPS y un botón dinámico para iniciar o detener el escaneo de dispositivos BLE.

La activity es totalmente dinámica, de forma que tanto las coordenadas como el número de datos enviados o los dispositivos escaneados son actualizados cada vez que alguno de ellos recibe alguna modificación.

2.4.2. Puntos clave de la aplicación

Durante el desarrollo de la aplicación se han encontrado tres puntos clave del sistema y los cuales son totalmente necesarios para que la aplicación cumpla los requerimientos necesarios: la obtención de las coordenadas GPS de la posición del voluntario, el escáner de dispositivos de BLE y el envío de datos del servidor.

2.4.2.1. Coordenadas GPS

El primer paso para poder obtener las coordenadas GPS del Smartphone del voluntario es pedir permiso al usuario de la app para poder utilizar la localización GPS del móvil, esto se realiza cuando se entra a la activity Gestión de datos por primera vez cuando se instala.

Una vez se clique en el botón de empezar servicio GPS, te redireccionará al menú de ajustes de GPS de Android para aclarar qué tipo de modo de ubicación quieres utilizar.

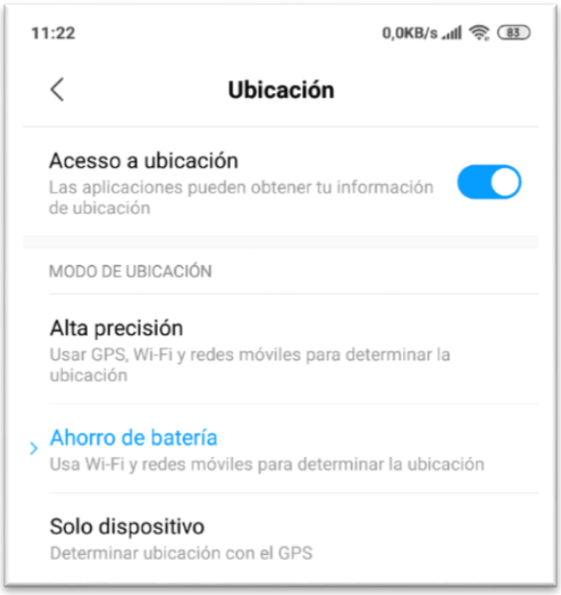

Fig 2.19 Tres modos diferentes de ubicación

Como se puede ver en la Figura 2.19, Android da la opción de obtener la ubicación vía tres modos: alta precisión, ahorro de batería y solo dispositivo. En el caso de nuestra aplicación, como necesitamos una gran precisión de las coordenadas para que después queden bien marcadas en el mapa, utilizamos el modo de Alta precisión. De esta manera, se utiliza para determinar la ubicación GPS, WIFI y redes móviles.

Otro de los requisitos que debe cumplir el servicio de obtención de las coordenadas GPS es que pueda trabajar en segundo plano o incluso con el móvil bloqueado, es por eso que se ha programado en Android como un Service.

Un Service es un componente de una aplicación que puede realizar operaciones de larga ejecución en segundo plano y que no proporciona una interfaz de usuario. Otro componente de la aplicación puede iniciar un servicio y continuará ejecutándose en segundo plano aunque el usuario cambie a otra aplicación.

```
locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, minTime: 3000, minDistance: 0,listener);
```

Figura 2.20 Código petición actualización coordenadas

En la Figura 2.20 se puede observar los parámetros para la petición de actualización de coordenadas, en este caso se ha establecido un tiempo mínimo de tres segundos y una distancia mínima de 0m. Esto significa que cada tres segundos o cada cambio de posición se solicitará unas nuevas coordenadas.

Cada vez que el listener del Service GPS detecte un cambio o actualización de coordenadas, el Service enviará vía broadcast a la activity las nuevas coordenadas y estas serán actualizadas en los textviews de Latitud y Longitud. Este proceso se puede observar en la siguiente Figura 2.21.

```
listener = new LocationListener() {
    @Override
    public void onLocationChanged(Location location) {//Función en la que entra el listener cuando detecta un cambio
        //Se crea el intent, se formatea la información necesaria y se envía en forma de broadcast a la activity que ha solicitado el servicio
        Intent i = new Intent( action: "location_update");
        i.putExtra( name: "coordinates", value: location.getLongitude()+" "+location.getLatitude());//Cogemos la información de longitud y latitud
        sendBroadcast(i);
```

Figura 2.21 Código de envío coordenadas vía broadcast a la activity principal

2.4.2.2. Escáner del Beacon BLE

El escaneado de los Beacon BLE es el proceso más fundamental de todo el sistema. Al principio del desarrollo, se utilizó la librería pública de Android dedicada a Beacons BLE, pero se llegó a la conclusión de que no era la adecuada debido a que solo podría detectar beacons de tipo iBeacon o de tipo Eddystone.

Visto que el objetivo era poder detectar cualquier tipo de beacon BLE, se pasó a utilizar una librería general de Android dedicada a Bluetooth (BluetoothManager). De esta manera la aplicación detecta cualquier tipo de dispositivo BLE, como se puede ver en la Figura 2.18 (Derecha), la app ha detectado dos tipos de dispositivos y muestra en el textview su nombre y potencia.

La función de escáner trabaja en un thread diferente para que la aplicación no se bloquee mientras está escaneando. Y es cuando detecta un dispositivo que llama a la función callback para procesar los datos del beacon escaneado.

```
public void startScanning() {//Función donde se inicia escáner Bluetooth
    System.out.println("start scanning");
    peripheralTextView.setText("");
    startScanningButton.setVisibility(View.INVISIBLE);//Ponemos el botón iniciar escáner en invisible
    stopScanningButton.setVisibility(View.VISIBLE);//Ponemos el botón stop escáner en visible
    AsyncTask.execute(new Runnable() {//Ejecutamos el scan en una asyntask para que corra en otro thread
        @Override
        public void run() {
            btScanner.startScan(leScanCallback);
        }
    });
}
```

Fig 2.22 Código de la función “Iniciar Escáner”

Como se puede observar en el extracto del código de la Figura 2.22, el escáner se inicia en una AsynTask para que corra dentro de otro thread. Y es en la sentencia de StartScan donde indica el nombre del callback al que acudir en el momento en el que detecta un dispositivo.

2.4.2.3. Envío de datos al servidor

El envío de datos al servidor se produce en el momento en el que el escáner detecta un beacon con un nombre siguiendo la siguiente estructura, donde las “X” serían sustituidas por el número de identificador del corredor:

CorredorID-XXXXXX

A continuación se muestra el código de la función Callback del escáner en la cual se revisa que el nombre del dispositivo siga la estructura que se requiere.

```
private ScanCallback leScanCallback = onScanResult(callbackType, result) → {//El objeto result contiene toda la info del dispositivo escaneado
    peripheralTextView.append("Device Name: " + result.getDevice().getName() + " rssi: " + result.getRssi() + "\n");//Obtenemos el nombre y la potencia del dispositivo escaneado
    try {
        //Los beacons de las carrera tienen este formato: CorredorID-000001
        //Comprovamos que la primera parte de la string se corresponde, para en ese caso, enviar el dato
        String idcor=String.valueOf(result.getDevice().getName());
        String[] idcorseparated = idcor.split( regex: "-");
        if (idcorseparated[0].equals("CorredorID")){
            enviardato(String.valueOf(result.getDevice().getName()));//llamamos a la función para enviar datos al server dando como parametro el nombre dle dispositivo
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
    // auto scroll para el text view
    final int scrollAmount = peripheralTextView.getLayout().getLineTop(peripheralTextView.getLineCount()) - peripheralTextView.getHeight();
    // Si no es necesatio el scroll, scrollAmount será <=0
    if (scrollAmount > 0)
        peripheralTextView.scrollTo( x: 0, scrollAmount);
};
```

Fig 2.23 Código de la función “callback” del escáner

Como se ve en la Figura 2.23 el callback recibe como parámetro un objeto ScanResult, el cual utilizando la librería mencionada anteriormente, nos permite obtener su nombre y su potencia con los comandos siguientes:

Nombre → result.getDevice().getName();

Potencia → result.getDevice().getRssi();

Una vez se obtiene el nombre y se comprueba su correcta estructura, este se pasa como parámetro a la función de envío de datos.

Para evitar la duplicidad de datos y siguiendo la lógica de que un voluntario solo debe recoger un dato de un mismo corredor, la aplicación tiene una memoria de corredores de los cuales ya se ha enviado datos. De esta manera, una vez se ha confirmado que el beacon detectado lleva la estructura correcta y además se ha comprobado que no se ha enviado anteriormente, se procede a enviarlo y además se añade a la memoria.

El proceso de envío de datos a la base de datos se hace con la librería Volley. Volley es una librería desarrollada por Google para optimizar el envío de peticiones HTTP desde las aplicaciones Android hacia servidores externos.

Para poder insertar datos en la base de datos es necesario hacerlo con un comando MySQL, el cual se hace desde un fichero PHP que está almacenado en el servidor. Y es para acceder a este fichero el por qué utilizamos el proceso Volley.

Los datos necesarios que se han de enviar desde la aplicación son el usuario del voluntario, latitud, longitud, nombre de la carrera y el ID de corredor. Toda esta información se mapea y se envía mediante Volley al fichero PHP del server que lo procesa y lo guarda en la base de datos.

```
// Creamos la cola de Peticiones/Solicitudes
queue = Volley.newRequestQueue( context: this);
// Los parametros para el php, hacemos el mapping para poder pasarlo a JSON
map.put("unameV", Usuario);
map.put("latitud", latitud);
map.put("longitud", longitud);
map.put("nomcarrera", Nom_Carrera);
map.put("IDCorredor", corredorid); //Lo recibimos por BLE
// Creamos la variable url para que se edite automaticamente desde el fichero Strings
String url="http://"+getResources().getString(R.string.ip_server_Login)+"/InsertarDato_APP.php";
request = new JsonObjectRequest(
        Request.Method.POST, // El método request
        url,
        new JSONObject(map), // Los parametros para el PHP
        (response) → {
                Log.d( tag: "Response", String.valueOf(response));
                // Aquí parseamos la respuesta
                JSONObject myJson = response;
                String resp = null;
                try { //Comprovamos la respuesta recibida del server
                    resp = myJson.getString( name: "res");
                    if(resp.equals("OK")) {//Se ha realizado correctamente
                        //alertmessageLogin(1);
                    }
                } catch (JSONException e) {
                    e.printStackTrace();
                }
        },
        (error) → { // En caso de error con la conexon con el server entrariamos en esta parte del código
                //alertmessageLogin(4);
        });
// Ejecutamos la solicitud para obtener la información en formato JSON
queue.add(request);
```

Fig 2.24 Código de la función Volley

En la Figura 2.24 se puede ver el código de la función Volley, la cual envía los datos recogidos al servidor. Siguiendo un orden de lectura de arriba a abajo en la figura, primero vemos como se mapean todas las variables que son necesarios para guardar el dato correctamente.

Justo después del mapeo, se crea la String que servirá para indicar la dirección URL del fichero PHP dentro del servidor. Está String se modifica automáticamente cuando se cambia el valor de la dirección IP del servidor en el fichero String localizado en: /app /src /main /res /values.

Una vez creado la String de la dirección URL, se inicia la request con el método Post y usando JSONObject para parsear el mapa de variables creado anteriormente. Se envía la request a la dirección URL fijada y se queda a la espera de una respuesta del servidor, que será un “OK” si todo ha funcionado correctamente.

En la parte del servidor, en concreto el fichero PHP que ha sido llamado por la request se empieza a ejecutar. El primer paso es crear conexión con la base de datos y recoger los datos que vienen incluidos en el método Post.

```php
$con = mysqli_connect('localhost','root','@eETaC2018','rot_v2');
if (!$con) {
    die('Could not connect: ' . mysqli_error($con));
}

$post = json_decode(file_get_contents("php://input"), true);
 // Obtenemos la información del array del método Post

   $usuario = $post['unameV'];
   $latitud = $post['latitud'];
   $longitud = $post['longitud'];
   $idcorredor = $post['IDCorredor'];
   $Nom_Carrera = $post['nomcarrera'];
```

Fig 2.25 Código fichero PHP del servidor

Como se puede ver en la Figura 2.25, primero se hace la conexión con la base de datos. El comando utilizado tiene cuatro parámetros:

- Dirección de la base de datos: En este caso como la base de datos la proporción XAMPP pues se indica localhost.
- Usuario de la base de datos: Se indica el usuario predeterminado de la base de datos.
- Contraseña de la base de datos: Al usuario predeterminado de la base de datos se le ha asignado una contraseña para aumentar la seguridad.
- Nombre de la base de datos: En este caso el nombre de nuestra base de datos es rot_v2.

También podemos ver en la Figura 2.25 como se recogen los valores de las variables que se han enviado con el Post.

Una vez tenemos ya todos los datos y después de hacer alguna consulta a la base de datos para tener algún dato extra relacionado, procedemos a hacer el Insert del dato en la base de datos.

```
$sql="INSERT INTO `datos` (`ID_Dato`, `ID_Carrera_Corredor`, `Hora`,
`Latitud`, `Longitud`, `Distancia`) VALUES (NULL, $idcarreracorredor,
CURTIME(), $latitud, $longitud, $distanciacarrera)";

if ($con->query($sql) === TRUE) {
    $response="OK";
} else {
    $response="KO";
}
```

Fig 2.26 Sentencia Insert en sql

En la Figura 2.26 se puede ver la sentencia SQL con la que se hace el Insert en la base de datos. Se puede observar como para introducir la hora en la que se ha cogido el dato, se utiliza el comando CURTIME(). Este comando accede a la hora que tiene el PC donde se está ejecutando el servidor y la escribe en el formato que necesita la base de datos.

Siempre que se envía un dato, se guardan las coordenadas enviadas como unas coordenadas fijas para ese voluntario. De esta manera, en el caso que el próximo dato que se quiera enviar se pierda la cobertura GPS y no se tengan coordenadas, el dato será guardado con esas coordenadas fijas. En la Figura 2.27 se puede ver las sentencias SQL que actualizan las coordenadas (latitud y longitud).

```
$sql="UPDATE carreras_voluntarios SET Latitud_fija = $latitud WHERE
ID_Carrera_Voluntario =$idcarreravoluntario";
$sql="UPDATE carreras_voluntarios SET Longitud_fija = $longitud WHERE
ID_Carrera_Voluntario =$idcarreravoluntario";
```

Fig 2.27 Sentencia Update en sql

Una vez guardado correctamente se envía una contestación a la aplicación confirmando la buena recepción de los datos y está actualiza el textview de número de datos enviados.

2.5. Resultados y demostración

Para comprobar que el resultado del sistema es satisfactorio y cumple con todas las necesidades requeridas se han realizado dos demostraciones. Una que comprueba el funcionamiento de la posición GPS y otra comprueba el funcionamiento del escáner Bluetooth.

2.5.1. Demostración 1: Posición GPS

En esta demostración se busca comprobar el buen funcionamiento de la obtención de las coordenadas GPS desde la aplicación además del envío y procesados de datos en el servidor.

Para poder trabajar desde la calle con el servidor es necesario pasarlo de localhost a una dirección IP pública. Es por eso que gracias al departamento TSC de la EETAC, pudimos conectar el servidor a una dirección pública como se ve en la captura siguiente.

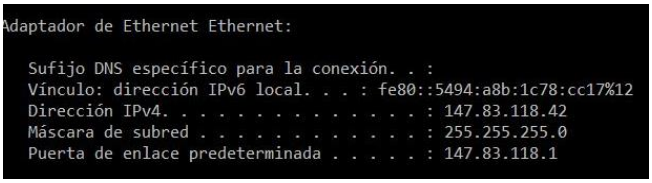

Fig 2.28 Captura ipconfig con IP pública

Una vez se tiene conexión desde la app con el servidor, ya se puede empezar con la demostración, la cual va a consistir en dar una vuelta al edificio C4 del campus del Baix Llobregat. A continuación se muestra la captura de la página web que muestra los resultados de la prueba teniendo en cuenta que se hizo con un solo corredor, es decir, con un solo dispositivo BLE.

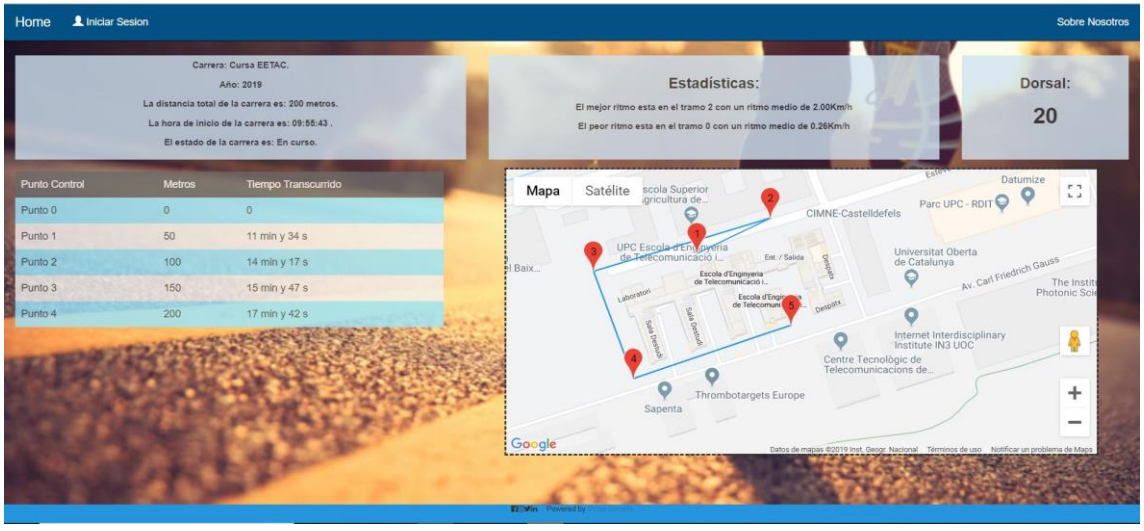

Fig 2.29 Resultados corredor demostración

Como se puede ver en la Figura 2.29, existen cinco marcadores dentro del mapa y en la tabla. El primero de los cuales se realizó dentro del laboratorio y por lo tanto la posición GPS no es del todo precisa.

El tiempo transcurrido en el desplazamiento entre el laboratorio y segundo punto son unos once minutos y medio y es a partir de aquí que empieza la vuelta al edificio. Como se puede ver perfectamente en el mapa, los marcadores 2, 3, 4 y 5 están perfectamente situados siguiendo la ruta seguida y además los tiempos transcurridos son totalmente coherentes teniendo en cuenta que el ritmo de movimiento fue lento.

También se puede comprobar el buen funcionamiento del apartado de estadísticas de la página mostrando como el tramo más lento el tramo entre el laboratorio y el segundo punto y el más rápido entre el punto 3 y 4, que viendo la tabla de tiempos, es el tramo en el que transcurre menos tiempo.

2.5.2. Demostración 2: Escáner Bluetooth

En esta demostración se busca ver la capacidad de gestión de datos que tiene la aplicación, es decir, si soportaría escanear y enviar un gran número de corredores en un corto espacio de tiempo.

Para poder hacer esta demostración, y gracias al departamento TSC de la universidad, se ha creado un sistema de 25 dispositivos BLE, como se puede ver en la Figura 2.30, alimentados por un pila, cada uno configurado con un número de identificador adecuado a lo especificado en el apartado 2.4.2.3.

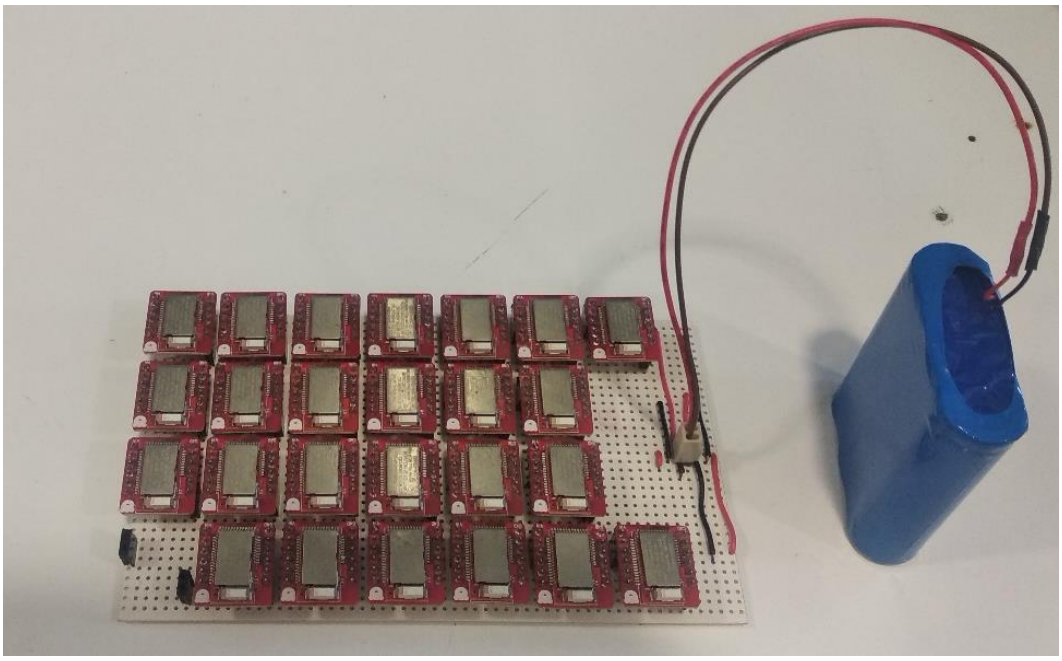

Fig 2.30 Modulos prueba másiva

El hecho de que los dispositivos utilizados para esta prueba trabajan con el estándar Bluetooth 4, provoca que el tiempo transcurrido en escanear todos los dispositivos sea mayor. Esto es debido a que los dispositivos no paran de transmitir en ningún momento y eso provoca que el escáner detecte y procese más de una vez el mismo dispositivo, en vez de escanear solo los que no han sido escaneados todavía.

Este inconveniente provoca lo que se puede ver en la Figura 2.31, donde los 11 primeros dispositivos se detectan en el mismo segundo mientras que entre el primer corredor detectado y el último existe una diferencia de 6 segundos.

Aun así se puede ver en la figura que se han detectado 25 dispositivos diferentes y todos en las mismas coordenadas.

ID_Dato	ID_Carrera_Corredor	Hora	Latitud	Longitud	Distancia
759	29	09:31:42	41.2755	1.9877	0
760	30	09:31:42	41.2755	1.9877	0
761	19	09:31:42	41.2755	1.9877	0
762	18	09:31:42	41.2755	1.9877	0
763	29	09:31:42	41.2755	1.9877	0
764	30	09:31:42	41.2755	1.9877	0
765	22	09:31:42	41.2755	1.9877	0
766	23	09:31:42	41.2755	1.9877	0
767	30	09:31:42	41.2755	1.9877	0
768	23	09:31:42	41.2755	1.9877	0
769	17	09:31:42	41.2755	1.9877	0
770	28	09:31:43	41.2755	1.9877	0
771	21	09:31:43	41.2755	1.9877	0
772	20	09:31:43	41.2755	1.9877	0
773	27	09:31:44	41.2755	1.9877	0
774	26	09:31:44	41.2755	1.9877	0
775	26	09:31:44	41.2755	1.9877	0
776	22	09:31:44	41.2755	1.9877	0
777	24	09:31:45	41.2755	1.9877	0
778	28	09:31:45	41.2755	1.9877	0
779	25	09:31:46	41.2755	1.9877	0
780	24	09:31:46	41.2755	1.9877	0
781	15	09:31:47	41.2755	1.9877	0
782	16	09:31:48	41.2755	1.9877	0
783	21	09:31:48	41.2755	1.9877	0

Fig 2.31 Tabla 25 datos BBDD

Como se ha explicado en el apartado 1.2.1.1 existen diferentes tipos de PDU, es decir de formas de trabajar con los dispositivos BLE. En este sistema se ha trabajado con el más común y simple ADV_NONCONN_IND además de utilizar el estándar de Bluetooth 4.

Este inconveniente de tiempo puede ser solucionado aplicando un par de cambios al sistema. Estos cambios serían: cambiar al estándar de Bluetooth 5 y utilizar el tipo de PDU ADV_SCAN_IND.

El uso de este tipo de PDU en el nuevo estándar permite que cuando el advertiser recibe el request enviado por el escáner y envía la información requerida, puede crear un evento que se dirige a capas superiores permitiendo que el dispositivo BLE deje de transmitir un intervalo de tiempo.

Este intervalo de tiempo sin transmitir permite que la probabilidad del escáner para detectar nuevos advertiser no detectados anteriormente aumente y de esta forma que el tiempo total de captura de todos los dispositivos BLE disminuya.

CAPÍTULO 3. EXPORTAR PROYECTO

El proceso de exportar el proyecto a otro ordenador se divide en dos puntos: exportar la aplicación y exportar el servidor.

Todo lo necesario para exportar el proyecto se encuentra en el siguiente link:

https://www.dropbox.com/sh/bcelzyeaguh9u6w/AAAM_NY1rdMMKN-9i3Ip1e3sa?dl=0

3.1. Exportar aplicación

El proceso de exportar es sencillo y solo basta con seguir unos pocos pasos que se detallan a continuación.

1) Acceder al link superior, entrar en la carpeta aplicación y descargar el RoT_app.zip.
2) Descomprimir el zip descargado en la carpeta que interese.
3) Abrir Android Studio.
4) File → New → Import Project
5) Seleccionar la carpeta con el nombre RoT y símbolo de Android.
6) Esperar a que se instalen todas las librerías i SDK necesarios.
7) Abrir el fichero String localizado en: /app /src /main /res /values.
8) Cambiar en el fichero Strings, la IP (no tocar el ":8080") del servidor que estemos utilizando de manera que cambiándolo solo ahí se cambia en todo la aplicación.

3.2. Exportar servidor

El proceso de exportar el servidor es bastante largo y se realiza siguiendo los siguientes pasos:

1) Instalar XAMPP [3].
2) Ir al XAMPP Control panel.
3) Clica en apache → config → Apache (httpd.conf).
4) Busca por "Listen 80" y sustitúyelo por "Listen 8080".
5) Busca por "ServerName localhost:80" y susituyelo por "ServerName localhost:8080".
6) Clica en apache → config → Apache (httpd-xampp.conf).
7) Busca por "**Alias /phpmyadmin "C:/xampp/phpMyAdmin/" AllowOverride AuthConfig Require local**" y sustitúyelo por "**Require all granted ErrorDocument 403 /error/XAMPP_FORBIDDEN.html.var"**.

8) Ir a xampp > config > clica en “service and port setting” → cambia el puerto Apache por 8080.
9) Si es necesario activar el permiso para las dos aplicaciones “http.exe” como se ve en la siguiente figura.

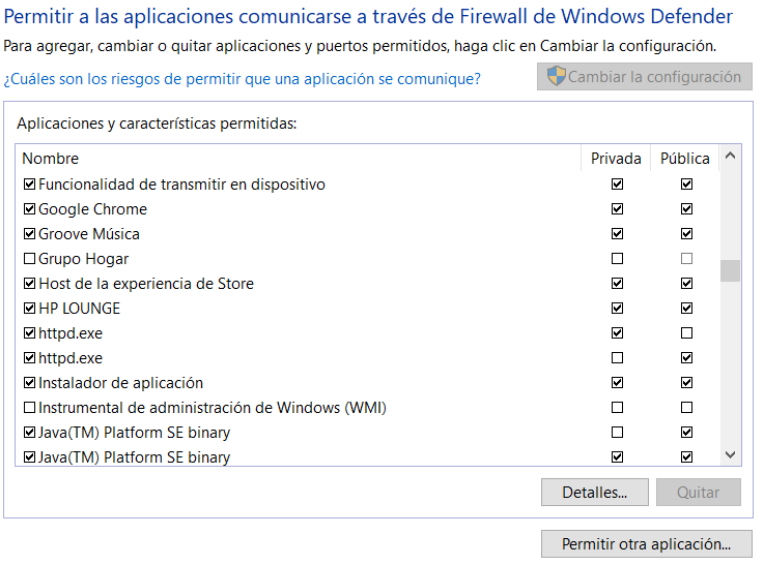

Fig 3.1 Configurador permiso Aplicaciones

10) Reinicia XAMPP.
11) Acceder al link superior, entrar en la carpeta servidor y descargar tanto el fichero de BBDD como todos los de htdocs.
12) Copiar todos los ficheros de la carpeta htdocs en el directorio de XAMPP carpeta htdocs.
13) Acceder al PhpMyAdmin desde el XAMPP Control Panel.
14) Crear base de datos con el nombre “rot_v2”.
15) Importar la base de datos desde el fichero que hemos descargado.

CAPÍTULO 4. CONCLUSIONES Y TRABAJO FUTURO

Hoy en día la tecnología más habitual para realizar el seguimiento de corredores en carreras es la basada en RFID. El objetivo final del trabajo era implementar una alternativa a este sistema utilizando la tecnología BLE, la cual es mucho más accesible y barata debido a su compatibilidad con la mayoría de Smartphones.

Para que todo el sistema fuera operativo, se han tenido que superar algunos obstáculos. Destacar el hecho de tener que detectar cualquier tipo de dispositivo BLE y no solo los beacons típicos como iBeacon o Eddystone. O el hecho de poder trabajar en segundo plano tanto con la actualización de coordenadas como de escáner de dispositivos.

Otra limitación inherente a este sistema es que el voluntario debe llevar activadas la ubicación y el Bluetooth en su dispositivo. A día de hoy, por la mala percepción que se tiene del consumo de batería, se ha implementado en la interfaz de la página de gestión de datos la posibilidad de activar o desactivar los servicios de escáner de dispositivos o actualización de coordenadas, con el objetivo de reducir el consumo de batería lo máximo posible. Otro contratiempo que se ha solucionado es el hecho de perder cobertura GPS el cual se ha solucionado con un sistema de coordenadas fijas.

También durante esta memoria se ha cubierto el objetivo de documentar la tecnología que utilizan estos dispositivos (ver Capitulo 1). La tecnología base en la que se basan los beacons ya es, a día de hoy, suficiente potente para apostar por ella.

Finalmente comentar que el objetivo del trabajo ya es una realidad. La plataforma de seguimiento de corredores en carreras populares ya cumple con todos los objetivos marcados y se han obtenido los resultados esperados como se ha podido comprobar en las diferentes demostraciones.

4.1. Trabajo Futuro

La aplicación desarrollada no deja de ser una buena base susceptible a mejora mediante la implementación de más servicios, algunos comentados ya en esta memoria como, la utilización de dispositivos BLE con la versión de Bluetooth 5 para que se desconecten un intervalo de tiempo una vez detectado un advertisement. En el caso que nos ha ocupado, sería interesante ampliar la cantidad de estadísticas que se le puede mostrar al usuario tales como desnivel total de la carrera o grafico de pulsaciones. Además también podría adaptarse la web a una aplicación para móviles y de esta manera ver los tiempos y estadísticas sin tener que utilizar el navegador. Otra posible mejora podría ser la comparación de estadísticas entre carreras para poder observar mejoras en la forma física.

CAPÍTULO 5. BIBLIOGRAFÍA

[1] Pérez Díaz de Cerio, David & Hernández, Ángela & Luis Valenzuela, Jose & Valdovinos, Antonio. (2017). Analytical and Experimental Performance Evaluation of BLE Neighbor Discovery Process Including Non-Idealities of Real Chipsets. Sensors. 17. 499. 10.3390/s17030499.

[2] Tosi, Jacopo & Taffoni, Fabrizio & Santacatterina, Marco & Sannino, Roberto & Formica, Domenico. (2017). Performance Evaluation of Bluetooth Low Energy: A Systematic Review. Sensors. 17. 2898. 10.3390/s17122898.

[3] Bluetooth Beacon, Beaconstac, último acceso: 09/07/2019
https://www.beaconstac.com/what-is-a-bluetooth-beacon

[4] Bluetooth Beacon vs RFID, Beaconstac, último acceso: 09/07/2019
https://blog.beaconstac.com/2015/10/rfid-vs-ibeacon-ble-technology/

[5] Especificaciones del dispositivo BLE, último acceso: 09/07/2019
https://os.mbed.com/platforms/RedBearLab-BLE-Nano/

[6] Mbed página web oficial, último acceso: 09/07/2019
https://www.mbed.com/en/

[7] Web oficial servidores apache - Xampp, último acceso: 09/07/2019
https://www.apachefriends.org/es/index.html

[8] Google Cloud Platform. Google, último acceso: 09/07/2019
https://cloud.google.com/?hl=es

[9] Front-end component library. Bootstrap, último acceso: 09/07/2019
https://getbootstrap.com/

[10] Android Studio. Android, último acceso: 09/07/2019
https://developer.android.com/studio/?gclid=CjwKCAjwuqfoBRAEEiwAZErCsqLKoJXC_ViyNV4oXW9UpPDY6bl42R93b9mYrlw8LwMhdQu0oD9odhoCFuoQAvD_BwE

[11] Link a todo el código tanto de la aplicación android y el servidor web. Dropbox, https://www.dropbox.com/sh/bcelzyeaguh9u6w/AAAM_NY1rdMMKN-9i3Ip1e3sa?dl=0

CAPÍTULO 6. ANEXO

6.1. Estructura tablas BBDD

- Tabla Voluntarios (4 campos):
    - ID_Voluntario → Campo único para cada voluntario
    - Email → Email del voluntario
    - Usuario → Usuario del voluntario para iniciar sesión
    - Password → Contraseña del voluntario para iniciar sesión.

- Tabla Carreras (8 campos):
    - ID_Carrera → Campo único para cada carrera
    - Nom_Carrera → Nombre de la carrera
    - Ciudad → Ciudad donde se hace la carrera
    - KM → Distancia de la carrera
    - Estado_Carrera → 0 indica que aún no ha empezado, 1 indica que está en curso y 2 indica que ya ha terminado.
    - Numero_Voluntarios → Número total de voluntarios que hay en la carrera
    - Hora_Fecha_Inicio → Hora y fecha exacta en la que empieza la carrera.
    - Año → Año en el que se hace la carrera

- Tabla Corredores (9 campos):
    - ID_Corredor → Campo único para cada corredor
    - Usuario → Usuario del corredor para iniciar sesión
    - Email → Email del corredor
    - Password → Contraseña del corredor para iniciar sesión.
    - Genero → Hombre o mujer
    - Nombre → Nombre del corredor
    - Apellidos → Apellidos del corredor
    - Localidad → Localidad del corredor
    - Fecha_Nacimiento → Fecha nacimiento del corredor

- Tabla Datos (6 campos):
    - ID_Dato → Campo único para cada dato
    - ID_Carrera_Corredor → Clave foránea para relacionar los datos con un único corredor en una carrera (Tabla Carreras_Corredores)
    - Hora → Hora en la que se recoge el dato
    - Latitud → Coordenada latitud

  - Longitud → Coordenada longitud
  - Distancia → Distancia desde el inicio de la carrera

- Tabla Carreras_voluntarios (7 campos):

  - ID_Carrera_Voluntario → Campo único que identifica y relaciona voluntario y carrera.
  - ID_Carrera → Clave foránea para relacionar las carreras
  - ID_Voluntario → Clave foránea para relacionar los voluntarios
  - Posicion_Voluntario → Dentro del número totales de voluntarios que hay en esa carrera en qué posición se sitúa.
  - Latitud_Fija → Latitud fija que se usará en caso de no tener buena cobertura gps.
  - Longitud_Fija → Longitud fija que se usará en caso de no tener buena cobertura gps.
  - Distancia_Respecto_Salida → Distancia a la que se sitúa el voluntario respecto la salida de la carrera.

- Tabla Carreras_corredores (4 campos):

  - ID_Carrera_Corredor → Campo único que identifica y relaciona corredor y carrera.
  - ID_Carrera → Clave foránea para relacionar los voluntarios
  - ID_Corredor → Clave foránea para relacionar los voluntarios
  - Dorsal → Dorsal que lleva el corredor en la carrera relacionada.

6.2. Interfaces páginas web

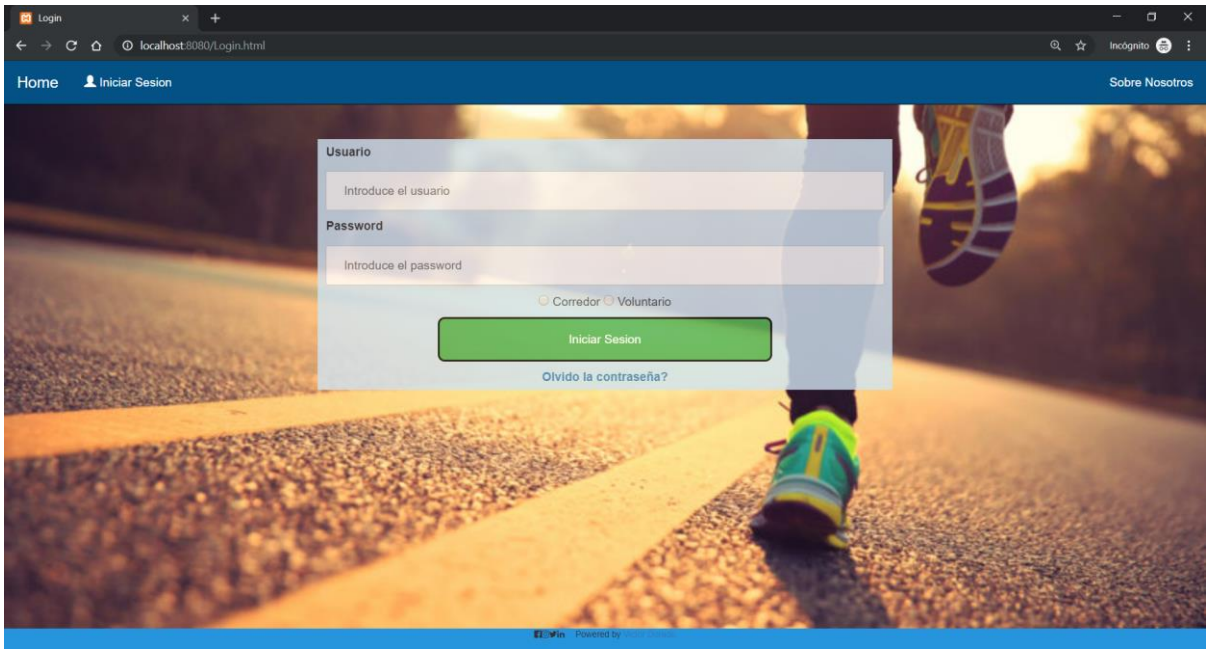

Fig 6.1 Página de inicio de sesión

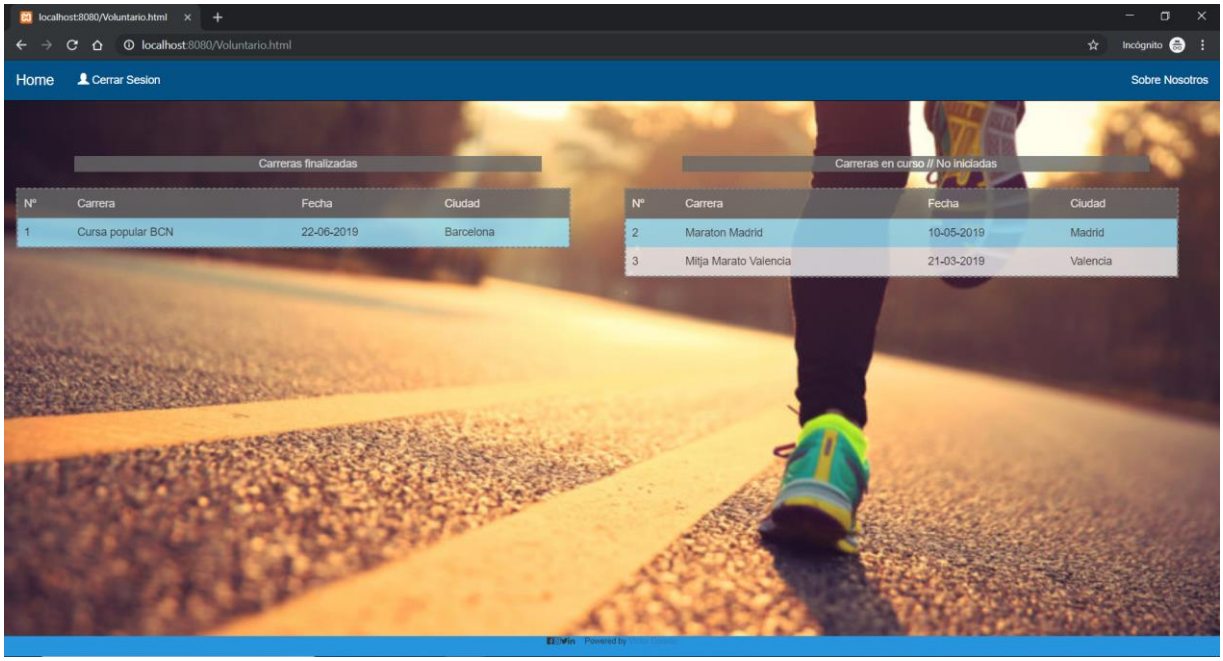

Fig 6.2 Página dedicada al voluntario

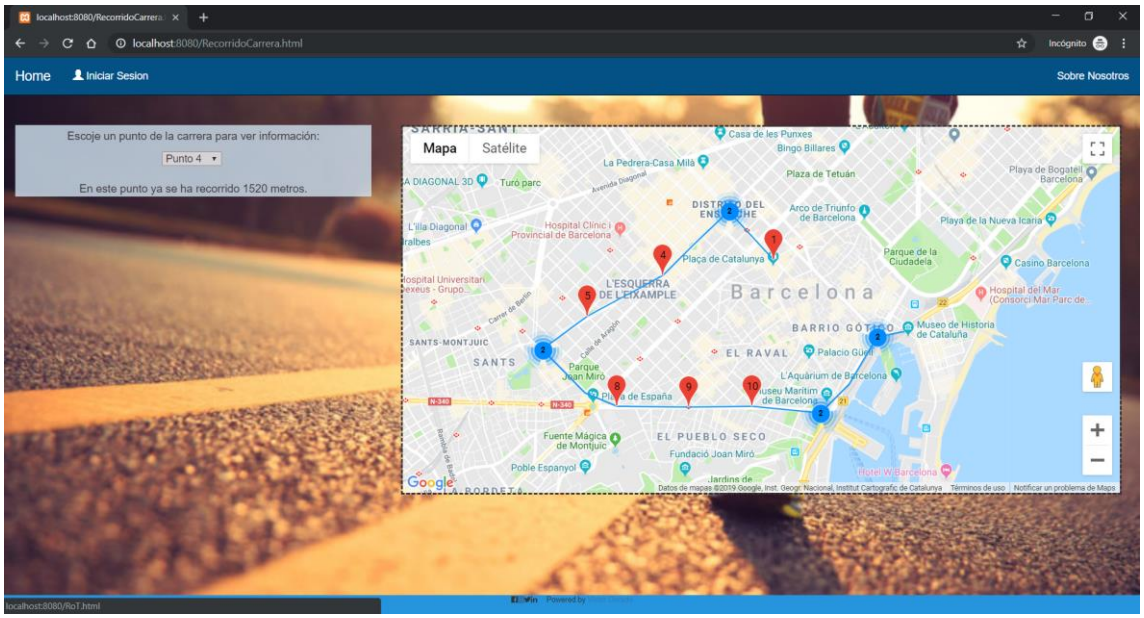

Fig 6.3 Página Recorridos

6.3. Código utilizado para la programación del dispositivo BLE

```
/* mbed Microcontroller Library
 * Copyright (c) 2006-2015 ARM Limited
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *     http://www.apache.org/licenses/LICENSE-2.0
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
#include "mbed.h"
#include "ble/services/iBeacon.h"

BLE ble;

void bleInitComplete(BLE::InitializationCompleteCallbackContext *params)
{
    BLE &ble          = params->ble;
    ble_error_t error = params->error;

    if (error != BLE_ERROR_NONE) {
```

```
        return;
    }

    uint8_t intent[] = {0x6A, 0x65, 0x6c, 0x6f, 0x75};

    uint8_t raw11[]={0x43, 0x6F, 0x72, 0x72, 0x65,
                     0x64, 0x6F, 0x72, 0x49, 0x44,
                     0x2D, 0x30, 0x30, 0x30, 0x30,
                     0x30, 0x31}; // CorredorID-000001

ble.accumulateAdvertisingPayload(GapAdvertisingData::COMPLETE_LOCAL_
NAME, raw11, sizeof(raw11));

ble.accumulateAdvertisingPayload(GapAdvertisingData::BREDR_NOT_SUPPO
RTED | GapAdvertisingData::LE_GENERAL_DISCOVERABLE);

    //Prepara el Scan Response

//ble.gap().accumulateScanResponse(GapAdvertisingData::COMPLETE_LOCA
L_NAME, intent, sizeof(intent));
    ble.gap().setAdvertisingInterval(100); /* in ms. */
    ble.gap().setTxPower(4); /**< in dBm (Valid values are -40, -20, -16, -12, -8, -
4, 0, 4) */

    // En el archivo nRF5xGap.cpp se pueden cambiar las channel_mask para
elegir los canales de advertising

ble.setAdvertisingType(GapAdvertisingParams::ADV_NON_CONNECTABLE_
UNDIRECTED); // Sin scan_request

//ble.setAdvertisingType(GapAdvertisingParams::ADV_SCANNABLE_UNDIRE
CTED); // Acepta scan_request
    ble.gap().startAdvertising();
}
int main(void)
{
    ble.init(bleInitComplete);

    /* SpinWait for initialization to complete. This is necessary because the
     * BLE object is used in the main loop below. */
    while (!ble.hasInitialized()) { /* spin loop */ }

    while (true) {
        ble.waitForEvent(); // allows or low power operation
    }
}
```

6.4. Código utilizado en la app y servidor

Todo el código tanto de la aplicación Android como del servidor web se puede encontrar en el Dropbox público. [7]